Master Science de la matière
École Normale Supérieure de Lyon
Université Claude Bernard Lyon I

Stage 2024–2025
Yacine Klikel
M1 Physique

Équilibrer la production intermittente et la consommation d'énergie dans les réseaux électriques par une approche jeux à champs moyens.

Résumé : Ce rapport étudie l'équilibre entre production renouvelable intermittente et consommation d'énergie dans les réseaux électriques. Nous utilisons une approche de théorie des jeux à champ moyen pour décrire le comportement des consommateurs et voir comment il influe sur le réseau. Une première partie présente l'état de l'art des modèles de réseaux et du cadre théorique des MFG. Après un aperçu de la physique des réseaux électriques, nous considérons un premier modèle qui donne une bonne intuition des résultats attendus dans les cas les plus complexes. La dernière parie de ce rapport explique comment implémenter un modèle plus raffiné et illustre le début de cette implémentation numérique.

Mots clefs : Jeux à champs moyens, Réseaux électriques, Systèmes complexes

Stage encadré par : **Denis Ullmo**
denis.ullmo@universite-paris-saclay.fr / tél. (+33) 1 69 15 74 76
Chercheur permanent au LPTMS
530 Rue André Rivière, 91400 Orsay

Laboratoire de Physique Théorique et Modèles Statistiques

25 juillet 2025

Table des matières

1 Introduction

Motivation : Le développement des énergies renouvelables et leur implémentation dans le réseau électrique pose de gros problèmes techniques aux fournisseurs d'électricité et de réseau. Ces nouvelles sources d'énergie rendent le réseau électrique plus compliqué à contrôler d'une part parce qu'on ne peut pas décider d'allumer ces sources lorsque la demande est très élevée sur le réseau comme on le ferait avec une centrale thermique par exemple, et d'autre part car leurs performances sont très dépendantes des aléas, météorologiques notamment. Ce projet s'inscrit dans une recherche d'autres moyens pour compenser cette perte de flexibilité. L'objectif de ce projet de recherche est de mieux comprendre le comportement des consommateurs et voir comment l'adaptation de leur comportement face à des prix variables pourrait aider à stabiliser le réseau électrique. Les consommateurs n'auraient pas eux-même à adapter leurs comportements mais possèderaient un automate chez eux qui déciderait de quand charger leurs voitures électriques et lancer les machines à laver par exemple. Il est même envisageables que les batteries, des voitures par exemples, puissent se décharger dans le réseau pour soulager un pic de demande. Les consommateurs complaisants seraient alors remerciés via leurs factures d'électricité.

On considère les clients d'un réseau électrique comme les agents d'un jeu d'optimisation. On entre ainsi en théorie des jeux et les « clients » du fournisseur d'électricité sont appelés des agents.

Dans un premier temps, les agents sont soit producteurs soit consommateurs et n'adaptent pas leur comportement. Pour raffiner le modèle, on peut considérer qu'ils adaptent leur production en fonction de la fréquence qu'ils mesurent dans le réseau électrique. Ils peuvent s'adapter immédiatement ou avec une latence τ et en moyennant ou non la fréquence mesurée sur une durée que l'on note T. Après avoir bien compris le comportement physique de ces réseaux, on permet aux agents d'adapter leur comportement à une réflexion stratégique individuelle. On voit ainsi l'apparition d'un équilibre de Nash.

1.1 Réseaux électriques

Cette partie s'appuie essentiellement sur un cours dispensé à CentralSupélec [1]. On y modélise le réseau électrique où l'$i^{\text{ème}}$ agent est relié aux autres et à la masse de la même manière qu'en figure 1.

On note P_{Di}, Q_{Di} les puissances active et réactive que le $i^{\text{ème}}$ agent consomme (la différence entre puissance active et passive est bien détaillée dans le cours de Centrale [1]). On introduit aussi P_{Di}, Q_{Di} les puissances actives et réactives fournies par l'agent. Dans la suite, on ne fera plus la distinction entre ce que l'agent produit et consomme : on considérera simplement $P_i^{\text{elec}} = P_{Di} - P_{Gi}$ la puissance active que l'agent i reçoit du réseau et $Q_i^{\text{elec}} = Q_{Di} - Q_{Gi}$ la puissance passive qu'il reçoit. Une loi des nœuds appliquée à $\underline{I}_i$ donne :

$$I_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} Y_{ijp} \cdot V_i + \sum_{\substack{j=1 \\ j\neq i}}^{N} Y_{ijs}(V_i - V_j) \tag{1}$$

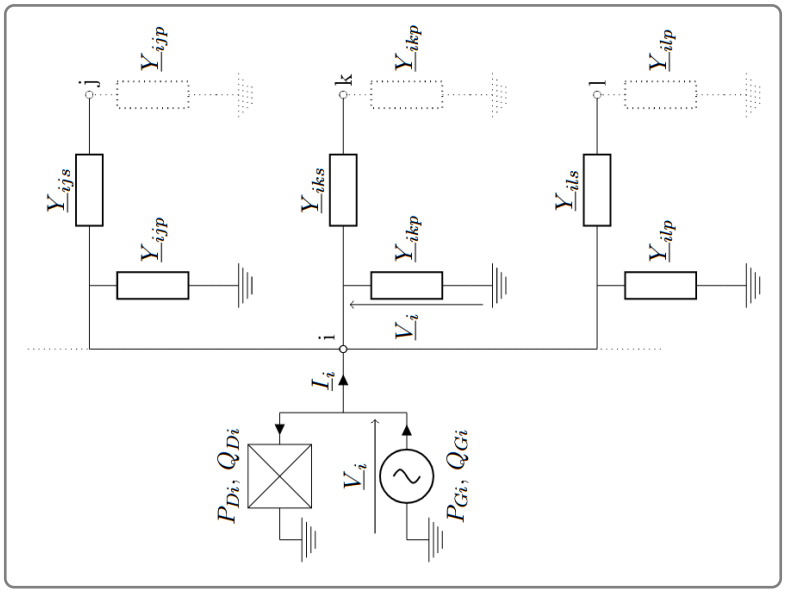


Figure 1 – Schéma général d'un noeud du réseau électrique (figure extraite du cours de CentralSupelec [1])

$$\text{On introduit } S_i = -P_i^{\text{elec}} - j \cdot Q_i^{\text{elec}}, \text{ et on a la relation : } \underline{S}_i = \underline{V}_i \cdot \underline{I}_i^* \tag{2}$$

$$\text{où l'on note : } \underline{V}_j = V_j e^{j\delta_j} \text{ et } \underline{Y}_{ij} = Y_{ij} \cdot e^{j\gamma_{ij}} \tag{3}$$

$$\text{On obtient alors les résultats : } P_i^{\text{elec}} = -\Sigma_{j=1}^{N} V_i V_j Y_{ij} \cos(\gamma_{ij} + \delta_j - \delta_i) \tag{4}$$

$$\text{et } Q_i^{\text{elec}} = \Sigma_{j=1}^{N} V_i V_j Y_{ij} \sin(\gamma_{ij} + \delta_j - \delta_i) \tag{5}$$

Ces $\underline{Y}_{ij}$ sont les impédances des lignes électriques entre les différents agents. Leur résistance est donc faible devant leur inductance donc on a $\gamma_{ij} \approx \frac{\pi}{2}$. D'où : $\cos(\gamma_{ij} + \delta_j - \delta_i) \approx \sin(\delta_j - \delta_i)$.

Usuellement, on modélise les agents comme des turbines. Ils ont ainsi une phase mécanique θ_i que l'on peut identifier à leur phase électrique : $\theta_i = \delta_i$. Ces agents sont aussi caractérisés par un moment m_i, un amortissement κ_i, et une puissance interne P_i^{int}. Cette puissance interne est positive lorsque l'agent apporte de l'énergie au réseau (par exemple via ses propres panneaux solaires) et négative lorsqu'il en consomme (par exemple en chauffant sa maison). En introduisant $\omega = \dot{\theta}$ la pulsation de l'agent i, on peut noter sa dynamique :

$$\begin{aligned} m_i \cdot \ddot{\theta}_i &= -\kappa \dot{\theta}_i + P_i^{\text{int}} + P_i^{\text{elec}} \\ &\approx -\kappa \dot{\theta}_i + P_i^{int} + \sum_{j=1}^{N} K_{ij} \sin(\theta_j - \theta_i) \end{aligned}$$

en notant K_{ij} une constante d'interaction entre les agents j et i.

On trouve ainsi exactement une équation de Kuramoto avec inertie dont le comportement a été largement étudié dans la littérature scientifique [4]. Une étude numérique du comportement de cette équation est tout de même proposée en partie 2.

1.2 Jeux à champs moyens dans le cas général

Cette sous-partie a pour objectif de présenter le cadre théorique des jeux à champs moyens (*Mean Field Games*, MFG) tel qu'il est établi dans la littérature. Plus pré-

cisément, il suit exactement le calcul proposé par Denis Ullmo dans le *Physics Reports* de 2019 [8]. Nous partons des équations différentielles stochastiques qui régissent la dynamique individuelle des agents pour arriver à l'équation de Hamilton-Jacobi-Bellman (HJB) et à celle de Fokker-Planck (également appelée équation de Kolmogorov avant)(rappelée en annexe A). Ce système couplé de deux équations renferme toute la complexité de la théorie des jeux à champs moyens.

Dans le cas général, on considère N agents caractérisés par leur variable d'état $\mathbf{X}^i \in \mathbb{R}^d,\ i \in \{1, .., N\}$. La variable d'état de chaque agent évolue suivant une équation différentielle stochastique :

$$\mathrm{d}\mathbf{X}_t^i = a_t^i \mathrm{d}t + \sigma \mathrm{d}\mathbf{W}_t^i \tag{6}$$

Dans cette équation, a_t^i représente le paramètre de contrôle de l'agent i. C'est via cette fonction qu'il détermine son état. $\mathrm{d}\mathbf{W}_t^i$ désigne l'incrément d'un mouvement brownien standard. On a essentiellement les résultats suivants pour de telles variables aléatoires :

$$\begin{aligned} \mathbb{E}\left[\mathrm{d}\mathbf{W}_t^i\right] &= 0 \\ \text{Pour } t \neq s, \quad \mathbb{E}\left[\mathrm{d}W_t^i\, \mathrm{d}W_s^j\right] &= 0 \\ \mathbb{E}[\mathrm{d}\mathbf{W}_t^i\, \mathrm{d}\mathbf{W}_t^j] &= \delta_{ij}\, \mathrm{d}t \end{aligned}$$

La répartition des N variables d'état $\mathbf{X^i}$ est décrite par une densité d'agent empirique dans l'espace des états :

$$m_t(x) = \frac{1}{N} \sum_j \delta(x - X^j(t)) \tag{7}$$

Ensuite, à chaque instant t chaque agent considère son coût personnel $c[\mathbf{a}](x,t)$ qui représente ce que l'agent espère payer entre l'instant t et la fin du jeu en adoptant la stratégie a sachant qu'il est dans l'état $x \in \mathbb{R}^d$ à l'instant t :

$$c_i[a^i](x,t) = \mathbb{E}\left\{ \int_t^T \left(L^i(\mathbf{X}_\tau^i, a_\tau^i) - \tilde{V}^i[m_\tau](\mathbf{X}_\tau^i) \right) d\tau + c_T^i(\mathbf{X}_T) \right\}$$

Dans ce coût, on peut lire le "*running cost*" dans l'intégrale : c'est le coût que l'agent paie à chaque instant. Dans cet article, comme dans la plupart de la littérature, ce coût se sépare en un premier terme $L^i(\mathbf{X}_\tau^i, a_\tau^i)$ dépendant du paramètre de contrôle individuel, et une seconde partie $\tilde{V}^i[m_\tau](\mathbf{X}_\tau^i)$ dépendant de l'état des autres joueurs. A cela s'ajoute un "*final cost*" qui dépend exclusivement de l'état final dans le formalisme le plus classique. Une simplification assez courante est de considérer que le coût ne dépend pas explicitement de l'agent i. C'est à dire que le coût subi par l'agent i dépend de son état et de son paramètre de contrôle mais un autre agent dans le même état subirait le même coût. Ce sera le cas dans tout ce rapport sauf dans la section (3) dans laquelle on n'utilise pas ce formalisme de toute façon. Pour continuer cet état de l'art, on fait donc cette hypothèse assez courante et on écrit ainsi le coût :

$$c[a^i](x,t) = \mathbb{E}\left\{ \int_t^T \left(L(\mathbf{X}_\tau^i, a_\tau^i) - \tilde{V}[m_\tau](\mathbf{X}_\tau^i) \right) d\tau + c_T(\mathbf{X}_T) \right\} \tag{8}$$

En pratique, à l'instant t, la stratégie des instants $\tau < t$ est déjà fixée et donc le $i^{\text{ème}}$ agent peut seulement choisir sur son paramètre de contrôle pour les instants futurs $\mathbf{a}\big|_{[t,T]}$ pour influer sur le coût. On définit le minimum $u(x,t)$ de ce coût suivant le contrôle :

$$u(x,t) = \min_{\mathbf{a}\big|_{[t,T]}} c[a^i](x,t) \tag{9}$$

C'est le coût minimal que l'agent peut espérer payer entre l'instant t et la fin du jeu en adoptant une stratégie optimale que l'on note $\mathbf{a}^*$ sous réserve d'existence.

L'hypothèse essentielle de la théorie des jeux à champs moyen est que pour N grand, $m(x,t)$ devient une densité déterministe. On peut appliquer HJB à u pour obtenir :

$$\begin{aligned} -\partial_t u &= H(x,\nabla u) + \frac{\sigma^2}{2}\Delta u - \tilde{V}[m](x) \\ u(x,T) &= c_T(x) \end{aligned} \tag{10}$$

$m(x,t)$ étant une densité déterministe, on peut lui appliquer l'équation de Fokker-Planck (cf annexe A) pour obtenir :

$$\begin{aligned} &\partial_t m(x,t) + \nabla \cdot \Big(m(x,t)a^*(x,t)\Big) - \frac{\sigma^2}{2}\Delta m(x,t) = 0 \\ &m_0(x) = m_0(x) \end{aligned} \tag{11}$$

Une dérivation de ces équations est proposée dans le *Physics Reports* [8] pour le cas général. Dans le cas particulier de notre modèle, la dérivation est faite en section (4).

Ces deux équations sont complètement intriquées mais HJB se résout en remontant le temps alors que FK se résout dans le sens d'écoulement du temps (elles sont respectivement dites *backward* et *forward*). Pour résoudre ces deux équations, la méthode la plus commune est d'itérer focker planck avec une stratégie initiale, puis faire plusieurs itérations de la boucle suivante :

— résoudre HJB
— injecter le résultat dans FK
— résoudre FK
— injecter le résultat dans HJB

En théorie, on peut arrêter la boucle lorsque la stratégie n'évolue plus d'une itération à l'autre.

Une autre méthode de résolution consiste à étudier le *régime stationnaire* : lorsque T est grand devant les temps caractéristiques d'évolution du système, on peut restreindre l'étude aux temps t loin des bornes 0 et T [8].

Il existe aussi des méthodes de résolution analytiques lorsque le coût est suffisamment simple. Les coûts quadratiques sont explicitement solvables par exemple [7].

2 Modélisation des réseaux électriques avec adaptations des consommateurs

Avant d'entrer dans le monde des jeux à champs moyens, il faut comprendre les dynamique résultante des *Swing Equations* que l'on a dérivé en partie 1.1. Cette partie vise donc à explorer les différents domaines de ces équations et comprendre leur comportement. On verra ensuite que l'évolution à champ moyen reste confinée dans une petite partie du domaine d'évolution de ces équations, dans une petite zone comprise dans le bassin d'attraction d'un point fixe.

2.1 Un agent couplé à un réseau infini

On considère un agent i couplé à un réseau dit *infini*. Un tel réseau est supposé suffisamment grand pour ne pas être affecté par la dynamique de l'agent i. Pour qu'un réseau comportant un grand nombre d'agents ait un tel comportement, il doit aussi être complet ou au moins fortement connecté [9]. En pratique, les réseaux que nous étudions ne sont pas *infinis*, mais leur comportement se rapproche de ceux des réseaux infinis proches des points fixes de fonctionnement.

Pour un tel réseau, on peut écrire la *Swing Equation* de l'agent i comme suit :

$$\frac{d}{dt}\begin{bmatrix}\theta_i\\ \omega_i\end{bmatrix} = \begin{bmatrix}\omega_i\\ -\frac{\kappa_i}{m_i}\theta_i + \frac{P_i}{m_i} - \frac{K_{i\infty}}{m_i}\sin(\theta_i - \Phi_\infty)\end{bmatrix}$$

où $K_{i\infty}$ est la constante de couplage entre l'agent i et le réseau infini, et Φ_∞ est la phase du réseau infini.

La dynamique de ce système est indépendante de la phase Φ_∞ du réseau infini tant que cette dernière varie lentement $\frac{m_i}{\kappa_i}$. On peut donc considérer que $\Phi_\infty = 0$ dans la suite :

$$\frac{d}{dt}\begin{bmatrix}\theta_i\\ \omega_i\end{bmatrix} = \begin{bmatrix}\omega_i\\ -\frac{\kappa_i}{m_i}\theta_i + \frac{P_i}{m_i} - \frac{K_{i\infty}}{m_i}\sin(\theta_i)\end{bmatrix} \tag{12}$$

On peut interpréter ce système comme l'évolution d'une particule (ou « balle ») soumise à un potentiel effectif $V(\theta_i) = -P\theta_i + 2K\cos(\theta_i)$ et à un terme de frottement proportionnel à $-\alpha\omega_i$. Sur la figure 2, on peut voir le potentiel effectif $V(\theta_i)$ pour une dissipation $\frac{\kappa_i}{m_i} = 1.25\ 2\pi$Hz (cette valeur particulière est choisie pour correspondre à l'étude de la sous-section (2.3)) et pour différentes valeurs de P :

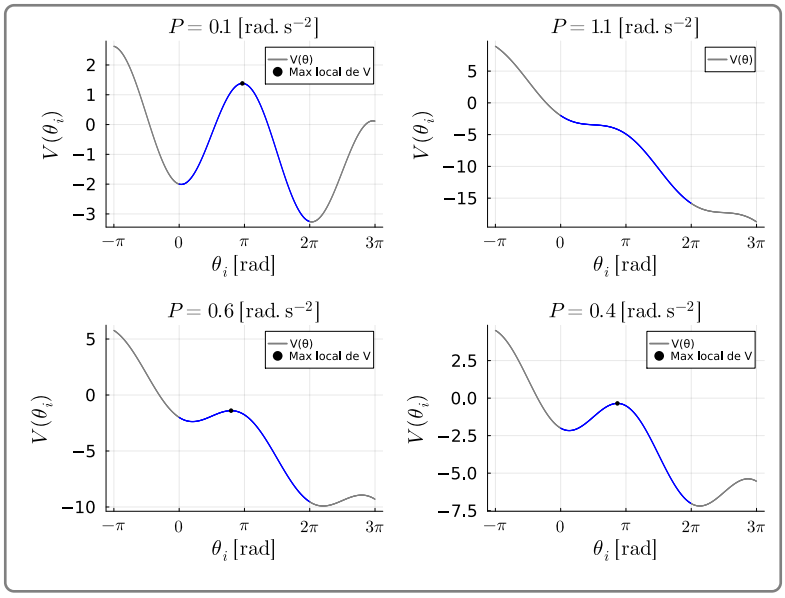


Figure 2 – Potentiel effectif $V(\theta_i)$ pour différentes valeurs de P et $\frac{\kappa_i}{m_i} = 1.25\ 2\pi$Hz *(Code en annexe C.1)*

Sur cette image, le potentiel semble être tronqué après une certaine valeur de θ_i. Il faut bien comprendre que, θ_i étant congrue modulo 2π, les abscices π et $-\pi$ correspondent en fait à la même valeur de θ_i et le système évolue donc suelemnt dans la partie bleue du potentiel. Cette figure permet de visualiser les 3 types de comportement détaillés ci-dessous. On aborde d'abord les cas limites avant de au cas intermédiaire :

- Lorsque P est suffisemment grand, comme en haut à droite de la figure 2, le potentiel est strictement décroissant donc la particule se déplace vers les valeurs de θ_i les plus hautes. Le système est dissipatif et θ_i est congru modulo 2π donc la particule finit par atteindre la trajectoire d'un cycle limite attracteur. Pour un agent couplé à un réseau infini, ce ype de comportement est celui que l'on observe lorsque les paramètres ne permettent pas à l'agent d'être en équilibre avec le réseau infini. Il se déphase donc de plusieurs tours par rapport au réseau infini, ce qui correspond à un fonctionnement hors de la plage de fonctionnement nominale du réseau.

- Lorsque P est très faible, comme en haut à gauche de la figure 2, les puits de potentiel sont trop profonds donc la particule n'arrive pas à conserver assez d'inertie pour en atteindre le sommet. Si on lache une particule dans un potentiel comme celui-ci, elle oscille autour de la position d'équilibre $\theta_i^{\text{éq}} = \arcsin\left(\frac{P_i}{K_{i\infty}}\right)$ de manière amortie. Elle finit ainsi par s'arrêter dans le puits de potentiel. Si elle est lancée vers la doite avec une vitesse élevée, elle peut parcourir plusieurs puits de suite mais en ayant une vitesse qui diminue à chaque puits qu'elle franchit. Elle finit par s'arrêter au fond d'un puits en $\theta = 0$. Pour un agent couplé à un réseau infini, ce type de comportement est celui que l'on observe lorsque les paramètres sont tels que l'agent est en équilibre avec le réseau infini.
- Lorsque P est entre les deux, comme sur les graphiques en bas de la figure 2, si la particule est lancée vers la droite ($\omega > 0$) avec suffisemment de vitesse, elle pourra atteindre le sommet du puits de potentiel puis elle emagasinera suffisemment d'énergie cinétique en descendant pour atteindre le sommet du puits suivant. Elle peut ainsi atteindre un cycle attracteur similaire au premier cas. On comprend cependant que si la particule est lancée avec une vitesse positive trop faible, elle restera confinée dans le puits de potentiel. Si elle est lancée vers la gauche ($\omega < 0$), elle remontera le potentiel jusqu'à faire demi-tour une première fois. La particule peut monter plusieurs puits de potentiel avant de faire demi-tour. Lors de ce premier demi-tour, si elle est assez proche du maximum local, elle aura suffisement de vitesse pour atteindre le sommet du puits suivant et entrer dans le cycle attracteur. Sinon, elle finit par s'arrêter dans le puits de potentiel le plus proche et atteindra ainsi le point fixe attracteur $\theta_i^{\text{eq}} = \arcsin\left(\frac{P_i}{K_{i\infty}}\right)$.

On en déduit que dans le premier cas, l'agent est en déséquilibre avec le réseau infini quelquesoient les conditions initiales ; tandisque dans le deuxième cas, l'agent est toujours en équilibre avec le réseau infini. Dans le troisième cas, l'agent peut être en équilibre ou non avec le réseau infini selon les conditions initiales. Il est donc intéressant de tracer un diagramme de phase pour visualiser les conditions initiales qui permettent à l'agent d'être en équilibre avec le réseau infini dans ce cas. On voit en figure 3 le diagramme de phase pour les conditions initiales $\theta_i(0) \in [-\pi, \pi]$ et $\omega_i(0) \in [-30, 30] rad/s$.

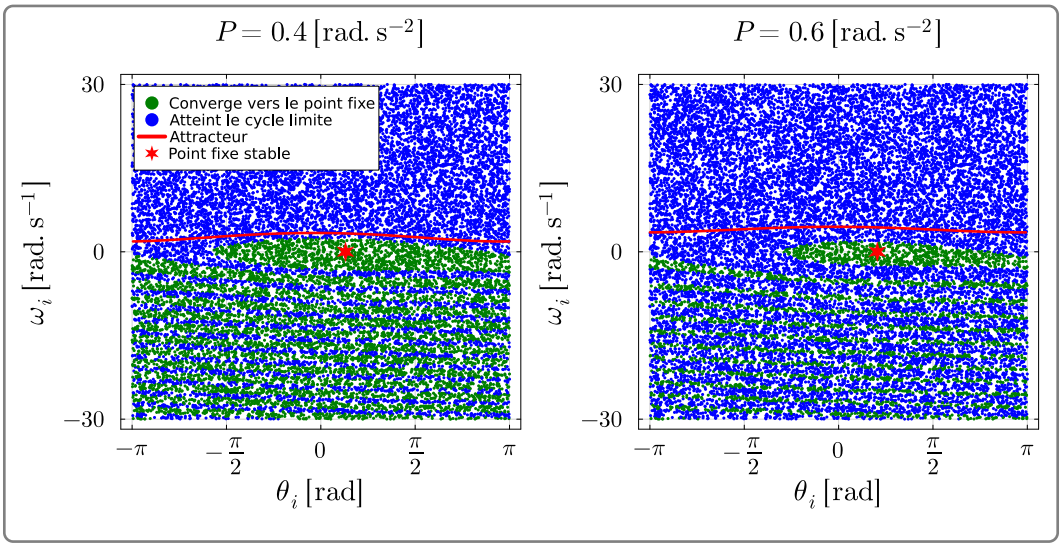


Figure 3 – Diagramme de phase pour un agent couplé à un réseau infini avec $\frac{\kappa_i}{m_i} = 1.25$ rad.s^{-1} et $K_{i\infty} = 8$ rad.s^{-1} *(Code en annexe C.1)*

Sur cette figure, les conditions initiales avec $\omega_i(0) > 0$ tendent vers le cycle attracteur

(trajectoire rouge), tandis que pour celles avec $\omega_i(0) < 0$, certaines tendent vers le cycle attracteur et d'autres vers le point fixe attracteur (étoile rouge).

Dans la suite, on veut savoir quels paramètres permettent d'avoir un équilibre stable global ou au moins un double équilibre comme dans mais avec un bassin d'attraction (zone verte sur la figure 3) suffisamment étendu pour être sur de rester dedans. Faire cela revient à tracer le diagramme de stabilité du sysème en fonction des paramètres αet$P$ pour un K donné. La zone d'existence du point fixe est assez simple à tracer, il suffit de vérifier que le potentiel effectif $V(\theta_i)$ admette un minimum local. Pour repérer l'existence du cycle attracteur, on remarque que la condition suivante est nécessaire et suffisante pour l'existence d'un cycle attracteur :

Le cycle attracteur existe si et seulement si : Une trajectoire partant des conditions initiales $\theta_i(0) = \text{argmax}_{\text{local}} V(\theta_i)$ et $\omega_i(0) > 0$ atteint le cycle attracteur et est encore en mouvemnt aux temps longs devant l'évolution du système.

Ainsi, pour differentes valeurs de P et α, on sait tester l'existence du point fixe et du cycle attracteur. Cela donne la figure 4.

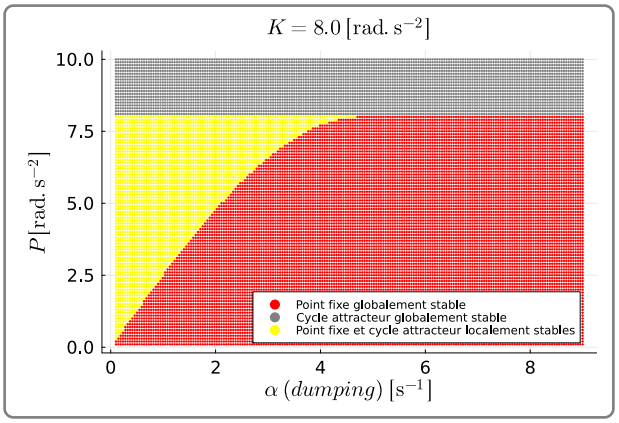


Figure 4 – Diagramme de stabilité du système pour différentes valeurs de P et α pour $K_{i\infty} = 8\ 2\pi\text{Hz}$ *(Code en annexe C.1)*

2.2 Réseau à 2 agents

Cette sous section monte que ce modèle permet aussi de décrire le comportement d'un réseau à 2 agents sans hypothèse. On sait que la dynamique d'un réseau à 2 agents est décrite par deux *Swing Equations* (1.1) :

$$\frac{d}{dt}\begin{bmatrix}\theta_1\\ \theta_2\\ \omega_1\\ \omega_2\end{bmatrix} = \begin{bmatrix}\omega_1\\ \omega_2\\ -\dfrac{\kappa_1}{m_1}\theta_1 + \dfrac{P_1}{m_1} + \dfrac{K_{12}}{m_1}\sin(\theta_2-\theta_1)\\ -\dfrac{\kappa_2}{m_2}\theta_2 + \dfrac{P_2}{m_2} + \dfrac{K_{21}}{m_2}\sin(\theta_1-\theta_2)\end{bmatrix} \tag{13}$$

L'interaction entre 2 agents couplés est ainsi décrit par par une équation différentielle vectorielle du premier ordre non linéaire. Ces equations différentielle regroupent l'interaction entre les agents et l'évolution de la valeur moyenne Φ du réseau. On peut réécrire ce système en utilisant les différences de phase et de vitesse angulaire entre les agents, notées $\Delta\theta = \theta_2 - \theta_1$ et $\Delta\omega = \omega_2 - \omega_1$ et la phase moyenne $\Phi = \frac{\theta_1+\theta_2}{2}$. On trouve ainsi 2 équations sur $(\Phi,\ \dot{\Phi})$ qui ne nous intéressent pas, et 2 équations sur $(\Delta\theta,\ \Delta\omega)$ que voici :

$$\frac{d}{dt}\begin{bmatrix}\Delta\theta\\ \Delta\omega\end{bmatrix} = \begin{bmatrix}\Delta\omega\\ 2P - \alpha\Delta\omega - 2K\cdot\sin(\Delta\theta)\end{bmatrix} \tag{14}$$

où on a noté $P = \frac{P_{10}-P_{20}}{2m}$, $K = \frac{K_{12}}{m}$ et $\alpha = \frac{\kappa_1+\kappa_2}{m}$.

On observe que ces équations décrivant un réseau à 2 agents s'identifient parfaitement à celles d'un agent couplé à un réseau infini. L'étude de la dynamique de ce système est donc parfaitement similaire à celle d'un réseau infini.

2.3 Réaction simple de l'agent

Que l'on considère un réseau à 2 agents ou un agent couplé à un réseau infini, on comprend ainsi bien la dynamique du système lorsque P est constant dans le temps. Voyons maintenant quel effet a l'adaptation de la puissance fournie par les agents.

Considérer une adaptation linéaire immédiate $P \to P - \gamma\omega$ est exactement équivalent à l'étude de la sous-section (2.1) en prenant un amortissement effectif $\alpha_{\text{eff}} = \alpha + \gamma > \alpha$. Cet amortissement plus grand signifie que les bassins d'attraction sont plus grands et on contrôle mieux la dynamique du système.

Pour raffiner le modèle, on peut considérer que cette adaptation s'effectue avec un retard τ, correspondant au temps de réaction de l'agent en s'inspirant de l'article [5] :

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \omega\end{bmatrix} = F(\begin{bmatrix}\theta\\ \omega\end{bmatrix}) = \begin{bmatrix}\omega\\ 2P - \alpha\omega - 2K\cdot\sin(\theta) - \gamma\omega_\tau\end{bmatrix} \tag{15}$$

$$\text{en introsuisant : } \omega_\tau(t) = \omega(t-\tau) \tag{16}$$

$$\text{et } P_i(\omega_i) = P_{i0} - \gamma\omega_{\tau i} \tag{17}$$

Ce système a un point fixe $X_{eq} = [\arcsin(P/K), 0]$ dont on veut étudier la stabilité.

Pour cela, on introduit la jacobienne J et la jacobienne retardée J_τ du système :

$$J = \begin{bmatrix}0 & 1\\ -2K\cos(\theta*) & -\alpha\end{bmatrix} \text{ et } J_\tau = \begin{bmatrix}0 & 0\\ 0 & -\gamma\end{bmatrix} \tag{18}$$

Pour étudier la dynamique proche du point fixe, notons $[\epsilon, \dot{\epsilon}]^{\mathrm{T}}$ l'écart à la valeur du point fixe. On a alors :

$$\frac{d}{dt}\begin{bmatrix}\epsilon\\ \dot{\epsilon}\end{bmatrix} = J\begin{bmatrix}\epsilon\\ \dot{\epsilon}\end{bmatrix} + J_\tau\begin{bmatrix}\epsilon_\tau\\ \dot{\epsilon_\tau}\end{bmatrix} \tag{19}$$

Pour étudier la stabilité du pont fixe, on cherche une solution sous la forme $\epsilon(t) = V\exp(\lambda t)$ avec $V \neq 0$. Cela revient à trouver λ tel que :

$$\lambda V \ = JV + J_\tau\exp(-\lambda\tau)V$$

Cela impose que λ soit solution de l'équation caractéristique :

$$\det(\lambda I - J - J_\tau\exp(-\lambda\tau)) = 0 \tag{20}$$

Cette équation caractéristique est transcendante, elle peut donc à priori avoir une infinité de solutions. Le point fixe est stable si $\max\mathrm{Re}(\lambda) < 0$. On peut donc calculer $\mathrm{Re}(\lambda)$ pour différentes valeurs de τ, simuler la dynamique du système et observer la corrélation entre les deux.

En pratique, cet exemple est étudié dans un article de Benjamin Shafer [5] où il observe la stabilité de ce système pour les paramètres :

$$P = 1\ \mathrm{s}^{-2}, \quad K = 8\ \mathrm{s}^{-2}, \quad \alpha = 0.1\ \mathrm{s}^{-1}, \quad \gamma = 0.25\ \mathrm{s}^{-1},$$

Avec ces paramètres, on peut tracer $F(\lambda) = \det(\lambda I - J - J_\tau \exp(-\lambda\tau))$ pour différentes valeurs de τ. On voit en figure 5 que pour $\tau = 5.0$ s, le système doit être stable car $\max \mathrm{Re}(\lambda) < 0$.

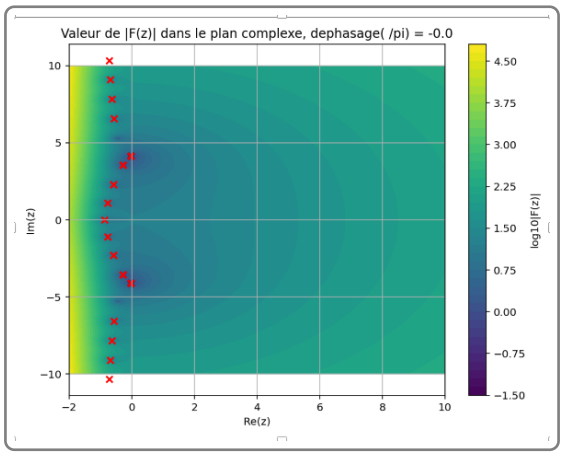
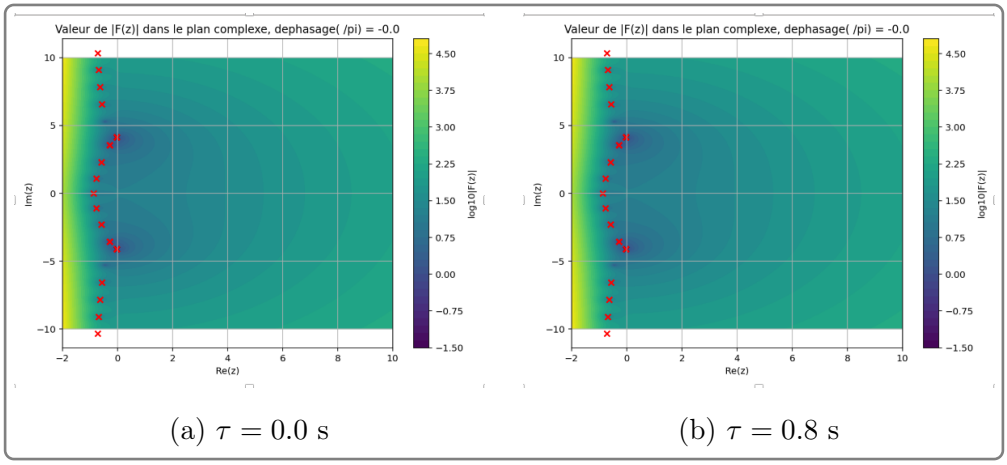


Figure 5 – Module de $F(\lambda)$ dans le plan complexe proche de lambda vérifiant max(re(lam)) parmi les zéros de F *(figure résultant d'une minimisation de fonction assez simple dont je n'ai pas mis le code en annexe par manque de place)*



(a) $\tau = 0.0$ s (b) $\tau = 0.8$ s

Figure 6 – Module de $F(\lambda)$ dans le plan complexe pour deux autres valeurs de τ.*(figure résultant d'une minimisation de fonction assez simple dont je n'ai pas mis le code en annexe par manque de place)*

On peut aussi tracer le module de $F(\lambda)$ dans le plan complexe pour $\tau = 0.8$ s et $\tau = 0.0$ s en figure 6. En observant les figures 5 et 6, on remarque ainsi que le temps de réaction des agents influe sur la stabilité du système. En pratique, lors de réalisations à grande échelle, il faudra donc veiller à ce que les bassins d'attraction des systèmes sans délai soient suffisamment grands pour ne pas risquer de perdre la stabilité en introduisant des petits délais τ inévitables.

3 MFG sans bruit

Dans ce premier modèle assez simple, on cherche à observer un équilibre de Nash en théorie des jeux à champ moyen sans ajouter de bruit dans les équations. On considère

une théorie des jeux à champ moyen dans laquelle chaque agent a un paramètre scalaire t plutôt qu'une fonction comme dans l'état de l'art 1.2.

Avec du bruit et un paramètre de contrôle fonction, on aurait obtenu des équations différentielles stochastiques (EDS). En enlevant le bruit, on obtient un résultat non stochastique et en passant à un contrôle scalaire, on obtient des équations non différentielles. On fait ces simplifications dans un premier temps pour se faire une première intuition des comportements résultants d'un équilibre de Nash. Dans la section suivante, on réintroduira les EDS pour s'intéresser à un modèle complet comme celui proposé en 1.2.

Pour ce modèle, on considère un nombre d'agents $N_{\mathbf{a}}$ représentant des consommateurs ; les clients d'un fournisseur d'électricité. Au cours d'une journée, chaque agent j a le comportement suivant :

- Il produit de l'énergie à une puissance $P_j(t)$ qu'il ne contrôle pas.
- Il consomme de l'énergie à une puissance $C_j(t)$, gaussienne centrée en un instant t_j qu'il contrôle (t_j est le paramètre de contrôle de l'agent j).
- Il souhaite consommer à un instant $\hat{t}_j$ qu'il ne contrôle pas ($\hat{t}_j$ se retrouve dans la fonction coût qu'il cherche à minimiser).
- Il paie un prix instantané $p(\langle\omega_i\rangle_i)$ (en Euros/Joule) pour la puissance qu'il consomme, où $\langle\omega_i\rangle_i$ est la moyenne de la vitesse angulaire des agents du réseau. Cette hypothèse sur le prix de l'électricité permet aux agents d'intéragir via un champ moyen. Il peut par exemple être interpréter comme un prééquilibre rapide dans 12. Dans ce cas, le champ moyen serait fait sur la dynamique du réseau et non sur la théorie des jeux.
- On définit ainsi sa fontion coût $\mathrm{Cout_j}(t_j, \hat{t}_j, \langle\omega\rangle)$ qui prend en compte le prix monétaire payé par l'agent, ainsi que le prix dit social qu'il paie en consommant de l'énergie à un instant différent de celui qu'il aurait souhaité.

3.1 Mise en équations

A priori, l'évolution de $\langle\omega_i\rangle_i$ résulte de l'évolution des $N_{\mathbf{a}}$ *Swing Equations* des agents du réseau (1.1). Pour rendre le problème accessible, on considère que tous les $N_{\mathbf{a}}$ agents ont les mêmes paramètres $(M, D, P_0, C_0, \sigma_p, \sigma_c)$. On peut alors sommer les *Swing Equations* pour obtenir l'évolution de $\langle\omega_i(t)\rangle_i$:

$$M \ \frac{d\,\langle\omega_i(t)\rangle_i}{dt} = \langle P_i(t)\rangle_i - \langle C_i(t)\rangle_i - D \ \langle\omega_i(t)\rangle_i \tag{21}$$

$$\text{où} \quad \langle P_i(t)\rangle_i = \frac{1}{N_{\mathbf{a}}} \sum_{i=1}^{N_{\mathbf{a}}} P_i(t) = P_j(t) \tag{22}$$

$$\text{et} \quad \langle C_i(t)\rangle_i = \frac{1}{N_{\mathbf{a}}} \sum_{i=1}^{N_{\mathbf{a}}} C_i(t) \tag{23}$$

Pour alléger les notations, nous omettons les indices i. De plus, la fonction coût que nous proposons dans ce modèle ne dépend pas explicitement de l'indice j ; nous ne l'indiquerons

donc plus dans la suite. Le j-ème agent est ainsi décrit par les équations suivantes :

$$P_j(t) = P_0 \exp\left(-\frac{(t-t_0)^2}{\sigma_p^2}\right) \tag{24}$$

$$C_j(t) = C_0 \exp\left(-\frac{(t-t_j)^2}{\sigma_c^2}\right) \tag{25}$$

$$\text{Cout}\left(t_j, \hat{t}_j, \langle\omega\rangle(t)\right) = \int_0^T \left[p\big(\langle\omega\rangle(t)\big) \cdot (P_j(t) - C_j(t))\right] dt + \lambda\left(t_j - \hat{t}_j\right)^2 \tag{26}$$

$$\langle\omega\rangle(t) = \frac{1}{M}\int_0^t \left[\exp\left(\frac{\tau - t}{M/D}\right) \cdot (\langle P(\tau)\rangle - \langle C(\tau)\rangle)\right] d\tau \tag{27}$$

On peut alors écrire le problème de minimisation du coût pour l'agent j comme suit :

$$t_j^* := \text{argmin}_{\text{t}_\text{j}}\left(\text{Cout}\left(t_j, \hat{t}_j, \langle\omega\rangle(t)\right)\right) \tag{28}$$

Où $\langle\omega\rangle(t)$ résulte elle même du comportement moyen des $N_\mathbf{a}$ agents du réseau. Chacun des $N_\mathbf{a}$ agents cherche donc à minimiser son coût en fonction de son paramètre de contrôle t_j et en prenant en compte le comportement des autres agents. Dans un tel modèle, les agents ont un comportement "égoiste" et cherchent à minimiser leur coût personnel sans prendre en compte le coût global du réseau.

Équilibre de Nash et équilibre sociétal : Le comportement collectif résultant peut être interprété comme un *équilibre de Nash*, dans lequel chaque agent choisit son paramètre de contrôle t_j afin de minimiser son propre coût, de manière individuelle et rationnelle, en tenant compte des décisions des autres. On atteint ainsi une situation stable où aucun agent n'a intérêt à modifier unilatéralement sa stratégie.

Dans une autre approche, on pourrait définir un coût global, ou *coût sociétal*, qui intègre non seulement les intérêts individuels des agents, mais aussi ceux du fournisseur d'énergie, ainsi que les impacts environnementaux liés à la consommation. Si les agents adoptaient une stratégie visant à minimiser ce coût global, le système évoluerait vers ce que l'on appelle un *équilibre sociétal*. Ce type d'équilibre pourrait mener à une meilleure performance collective et à une utilisation plus efficiente des ressources du réseau.

Pour pouvoir considérer un nombre d'agents très grands, on ne peut pas résoudre le problème de minimisation du coût pour chaque agent individuellement : on reformule le problème en terme de densités pour considérer un nombre d'agents qui peut être infini. On considère ainsi que les agents sont distribués selon une densité $\hat{\rho}(\hat{t})$ sur l'intervalle $[0, T]$. On cherche alors une fonction "stratégie" qui pour chaque instant $\hat{t}$, donne la stratégie résultante $t^* = \text{strat}_{\langle\omega\rangle}(\hat{t})$ à l'équilibre. Il faut comprendre ici que la stratégie adoptée par l'agent $\hat{t}$ dépend de $\langle\omega\rangle(t)$, donc la fonction $\text{strat}_{\langle\omega\rangle}(\hat{t})$ admet $\langle\omega\rangle$ comme paramètre. La donnée de la fonction $\text{strat}_{\langle\omega\rangle}(\hat{t})$ et de la densité $\hat{\rho}(\hat{t})$ permet de décrire le comportement collectif des agents : on introduit $\rho_c(t)$ la densité résultante de la stratégie des agents. $\rho_c(t)$ correspond à la densité d'agents qui ont choisi de consommer à l'instant t. Elle est intéressante car $\langle\omega\rangle$ découle directement de cette dernière, comme on le voit dans l'équation 32.

$$\text{strat}_{\langle\omega\rangle}(\hat{t}) = \text{argmin}_t\left(\text{Cout}\left(t, \hat{t}, x \to \langle\omega\rangle(x)\right)\right) \tag{29}$$

$$\langle\omega\rangle(t) = \frac{1}{M}\int_0^t \left[\exp\left(\frac{\tau - t}{M/D}\right) \cdot \left(\langle P(\tau)\rangle - \langle C(\tau)\rangle\right)\right] d\tau \tag{30}$$

$$\text{où} \quad \langle C(\tau)\rangle = \int_0^T C_0 \exp\left(-\frac{\left(\tau - \text{strat}_{\langle\omega\rangle}(x)\right)^2}{\sigma_c^2}\right) \cdot \hat{\rho}(x)dx \tag{31}$$

$$= \int_0^T C_0 \exp\left(-\frac{(\tau - x)^2}{\sigma_c^2}\right) \cdot \rho_c(x)dx \tag{32}$$

Les équations 29, 30, 31

- la stratégie individuelle à l'équilibre pour un agent donné (via une minimisation du coût) ;
- la dynamique agrégée du réseau à travers la fonction $\langle\omega\rangle(t)$;
- le couplage entre les décisions individuelles et l'effet global via la moyenne pondérée $\langle C(\tau)\rangle$.

Ce système est typique de ceux rencontrés dans les cadres dits de *Mean Field Games* (jeux à champ moyen), où un très grand nombre d'agents interagissent à travers une variable agrégée qu'ils influencent collectivement.

Pour résoudre ce genre de problème implicite et couplé, une approche numérique est généralement nécessaire. Elle consiste à itérer entre :

1. une étape de **mise à jour des stratégies** des agents : en résolvant le problème de minimisation pour chaque valeur de $\hat{t}$, étant donnée une estimation de $\langle\omega\rangle(t)$;
2. une étape de **mise à jour de** $\langle\omega\rangle(t)$: en recalculant cette grandeur à partir des stratégies obtenues et de la densité $\hat{\rho}(\hat{t})$;
3. vérifier s'il y a convergence d'un critère d'arrêt (souvent la stabilité de $\langle\omega\rangle(t)$ ou des stratégies individuelles).

3.2 Résolution numérique de l'équilibre de Nash

Dans cette section, nous présentons une méthode itérative pour approcher numériquement l'équilibre de Nash lorsque la densité d'état des agents, notée $\hat{\rho}(t)$, est discrétisée sur un maillage temporel uniforme.

3.2.1 Discrétisation des variables

- **Temps** : la plage $[0, T]$ est divisée en N intervalles de taille $\delta T = T/N$. On note $t_i = i\,\delta T$ pour $i = 0, \ldots, N-1$.

- **Densité d'état** : la densité continue $\hat{\rho}(t)$ devient un vecteur $\hat{\rho} = (\hat{\rho}_0, \ldots, \hat{\rho}_{N-1})^\top$, où

$$\hat{\rho}_i = \int_{i\,\delta T}^{(i+1)\,\delta T} \hat{\rho}(t)\,dt,$$

avec $\hat{\rho}_i \geq 0$ et $\sum_{i=0}^{N-1} \hat{\rho}_i = 1$.

- **Stratégies mixtes** : chaque groupe d'agents initialement localisé dans l'intervalle $[j\,\delta T, (j+1)\,\delta T]$ adopte une stratégie mixte décrite par un vecteur de probabilités

$$\text{strat}(j) = \left(s_{0j}, s_{1j}, \ldots, s_{N-1,j}\right),$$

tel que $s_{ij} \geq 0$ et $\sum_{j=0}^{N-1} s_{ij} = 1$. Nous regroupons toutes ces stratégies dans la matrice $\text{Strat} \in \mathbb{R}^{N\times N}$, dont l'élément $\text{Strat}_{ij} = s_{ij}$.

- **Densité de consommation** : la densité résultante des temps de consommation est le produit matriciel

$$\rho^c = \mathrm{Strat} \times \hat{\rho},$$

c'est-à-dire $\rho^c_j = \sum_{i=0}^{N-1} \mathrm{Strat}_{ji}\, \hat{\rho}_i$.

Les contraintes s'écrivent donc :

$$\begin{aligned} &\forall\, i, j, \quad 0 \leq \mathrm{Strat}_{ij} \leq 1, \\ &\forall\, j, \quad \sum_{i=0}^{N-1} \mathrm{Strat}_{ij} = 1, \\ &\rho^c = \mathrm{Strat} \times \hat{\rho}. \end{aligned}$$

3.2.2 Problème de convergence et cycle d'oscillation

Si l'on mettait à jour la stratégie de tous les agents simultanément en choisissant systématiquement le temps minimisant leur coût t^*, on observe un cycle attracteur de période 2 :

1. Les agents calculent t^* selon l'espérance de prix $\langle\omega\rangle$ issue de la stratégie actuelle.
2. Tous changent de stratégie pour optimiser leur coût.
3. L'actualisation de $\langle\omega\rangle$ inversée au pas précédent conduit les agents à bifurquer à nouveau, et ainsi de suite.

3.2.3 Algorithme de recherche d'équilibre avec amortissement

Pour briser ce cycle, nous introduisons un facteur d'amortissement $\alpha \in [0, 1]$. À chaque itération, seule une fraction α des agents adopte la nouvelle stratégie optimale, tandis que le reste conserve sa stratégie précédente.

Variables et initialisation - $\mathrm{Strat}^{(0)}$: matrice initiale de stratégies, par exemple tirée d'une loi de Dirichlet uniforme

$$\mathrm{Strat}^{(0)} = \mathrm{np.random.dirichlet(np.ones(N),}\ N).T,$$

ou choisie identique pour forcer chaque groupe à consommer à son $\hat{t}_j$ dès l'itération 0 : $\mathrm{Strat}^{(0)} = I_N$. - Critère de convergence : norme choisie sur $\mathrm{Strat}^{(k+1)} - \mathrm{Strat}^{(k)}$ inférieure à un seuil ε, ou nombre d'itérations maximal nb_iter.

Étape itérative Pour chaque itération $k = 0, 1, 2, \ldots$ jusqu'à convergence :

1. **Calcul de** $\langle\omega^{(k)}(t_i)\rangle$ pour $i = 0, \ldots, N-1$.
2. **Optimisation locale** : pour chaque groupe $j = 0, \ldots, N-1$, calculer le coût pour chaque t_i puis déterminer

$$i^*(j) = \arg\min_i \mathrm{Cout}\left(t_i, \hat{t}_j, \langle\omega^{(k)}\rangle\right).$$

3. **Construction de la matrice** $\mathrm{Argmins}^{(k)}$:

$$\mathrm{Argmins}^{(k)}_{ij} = \begin{cases} 1 & \text{si } i = i^*(j), \\ 0 & \text{sinon.} \end{cases}$$

4. **Mise à jour amortie** :

$$\mathrm{Strat}^{(k+1)} = (1-\alpha)\,\mathrm{Strat}^{(k)} + \alpha\,\mathrm{Argmins}^{(k)}.$$

5. **Vérification de convergence** : si $\|\mathrm{Strat}^{(k+1)} - \mathrm{Strat}^{(k)}\| < \varepsilon$ ou $k+1 = \mathrm{nb\_iter}$, arrêter l'algorithme.

3.3 Résultats

Cet algorithme donne un code clé en main dans lequel on choisit $\langle P(t)\rangle$, $\hat{\rho}(t)$ et les paramètres du problème M, D, ϵ, λ. Pour illustrer des résultats physique, considérons par exemple que la production $\langle P(t)\rangle$ soit une gaussienne centrée en midi, correspondant à de l'énergie solaire par exemple ou à des centrales qui fonctionneraient à plus haut régime lorsque tous les techniciens sont présents. Considérons que $\hat{\rho}(t)$ est répartie en double gaussienne autour de 8h et 20h du matin pour simuler des agents qui voudraient consommer de l'électricité en se préparant avant d'aller travailler et le soir en rentrant. On obtient ainsi les courbes vertes et oranges sur les figures ci-dessous.

Après convergence de l'algorithme, on obtient la stratégie résultante en vert sur la même figure :

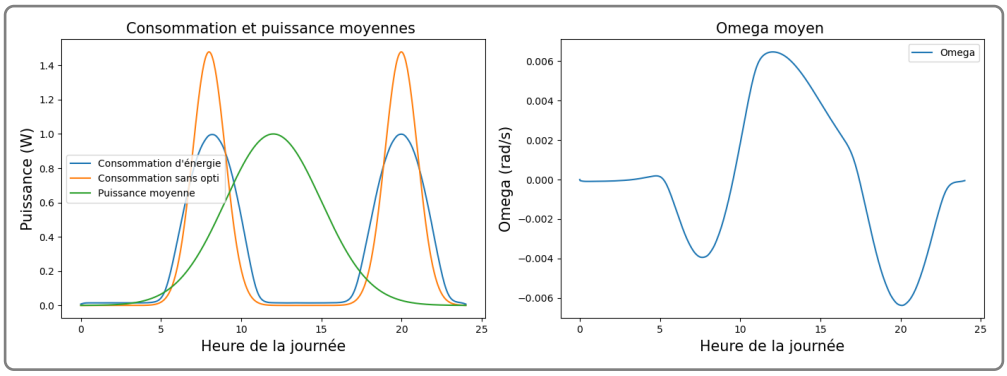


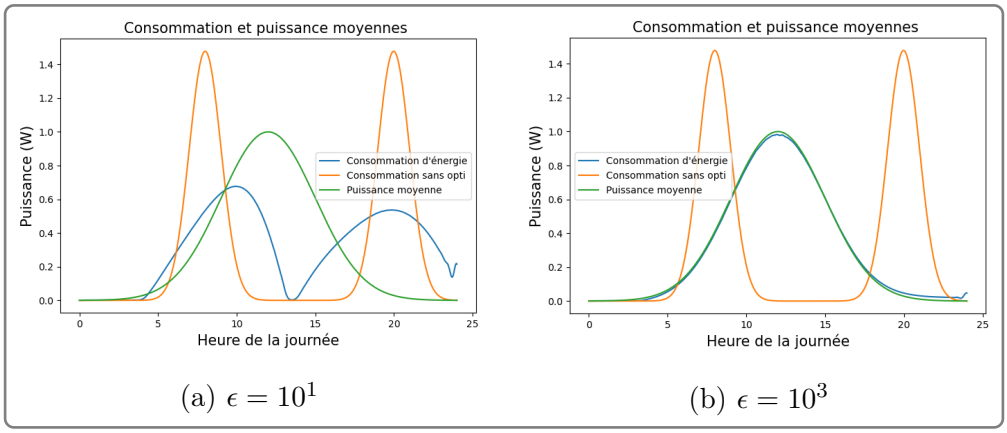
Figure 7 – Courbes résultante de l'algorithme détaillé en 3.2 pour les valeurs $(M, D, \epsilon, \lambda) = (10^3, \frac{M}{5}, 1, 1)$, dans les unités du système international. **À gauche**, on voit la production moyenne en vert, la consommation moyenne que l'on aurait si les habitants ne prenaient pas le prix de l'électricité en considération, la consommation moyenne lorsque les agents cherchent un compromis entre [consommer quand ils le veulent] et [payer l'électricité moins cher]. **À droite**, on voit le $\langle\omega\rangle(t)$ résultant de ce comportement compromis des agents. *(Code en annexe C.2)*

Ce code est satisfaisant car il permet de confirmer notre intuition sur le comportement des agents : ils consomment à l'instant voulu $\hat{t}$ ou le plus proche possible de cet instant tout en s'étalant un peu en direction des instants où les prix sont plus faibles. Cet étalement permet de diminuer la tension du réseau. Il est intéressant d'observer comment leur comportement dépend des différents paramètres du système. Pour cela, en partant des mêmes paramètres, on peut par exemple augmenter ϵ comme en figure 8. Cette variable dit de combien le prix du watt varie lorsque le réseau est en tension.

En augmentant ϵ, on considère un fournisseur d'électricité plus "punitif" lorsque le réseau est en tension et on observe que les agents font d'autant plus d'efforts pour éviter de consommer lorsque $\langle\omega\rangle(t)$ est faible. Ils adaptent ainsi leur comportement pour satisfaire les contraintes du fournisseur d'électricité.

De la même manière, des comportements intuitifs apparaissent lorsqu'on fait varier chacun des autres paramètres :

- Lorsque l'on augmente $\tau_{\text{réseau}} = M/D$ le temps caractéristique du réseau, $\langle\omega\rangle$ a une plus longue "mémoire" sur l'écart entre consommation en production. Il peut donc prendre des valeurs plus élevées si cet écart ne se compense pas. En conséquence, les agents doivent plus adapter leur comportement pour éviter les prix trop élevés. Une augmentation de $\tau_{\text{réseau}}$ est donc analogue à une augmentation de ϵ.
- Lorsque l'on diminue M, le réseau électrique a une inertie plus faible donc, de



(a) $\epsilon = 10^1$ (b) $\epsilon = 10^3$

Figure 8 – Code similaire à la figure précédente : seul ϵ change.

la même manière que lorsqu'on augmente le temps $\tau_{\text{réseau}} = M/D$, $\langle\omega\rangle$ devient sensible au comportement des agents. Cela oblige donc aussi les agents à adopter leur comportement. Une diminution de M est donc analogue à une augmentation de ϵ.

- Dans l'algorithme, λ et ϵ interviennent sous la forme $\frac{\lambda}{\epsilon}$ donc augmenter l'un est équivalent à diminuer l'autre.

4 Modèle avec bruit (3D)

4.1 Mise en équations

4.1.1 Paramètres du modèle

On considère un grand nombre d'agents, caractérisés par leur variable d'état aléatoire (θ, ω, E), où E représente l'énergie accumulée par l'agent. On devrait ainsi avoir N jeux de 3 équations différentielles stochastiques (EDS) :

$$\begin{aligned} d\theta_i &= \omega_i \cdot dt \\ d\omega_i &= \frac{1}{m}(-\kappa\omega_i + P_i + \frac{\lambda}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i)) \cdot dt + \frac{2D}{m} \cdot dW_i^1 \\ dE_i &= P_i \cdot dt + \sigma_i \cdot dW_i^2 \end{aligned}$$

Pour pouvoir passer à la limite $N \to \infty$, on doit introduire :

- $r\exp(i\Psi) := \frac{1}{N}\sum_{j=1}^{N}\exp(i\theta_j)$ le paramètre d'ordre usuelle
- $\rho_t(\theta, \omega, E)$ la densité d'état des agents à l'instant t

On peut alors réécrire les EDS de la manière suivante pour un nombre d'agents infini :

$$\begin{aligned} d\theta &= \omega \cdot dt \\ d\omega &= \frac{1}{m}(-\kappa\omega + P + \lambda \cdot r\sin(\Psi - \theta)) \cdot dt + \frac{2D}{m} \cdot dW^1 \\ dE &= P \cdot dt + \sigma \cdot dW^2 \end{aligned} \tag{33}$$

On définit ensuite un coût $\text{Cout}[P](\theta(t), \omega(t), E(t); t)$:

$$\text{Cout}[P](\theta(t), \omega(t), E(t);\, t) = \mathbb{E}\int_t^T \left[\eta\left(E(\tau) - \hat{E}(\tau)\right)^2 - P(\tau)\cdot p(\omega(\tau)) + \frac{1}{2}\mu\left(P(\tau)\right)^2\right] d\tau \tag{34}$$

Où :

- $\hat{E}$ est une fonction consommation cible identique pour tous les agents.
- p est le prix instantané de l'énergie, dépendant exclusivement de ω.
- μ est un paramètre de régulation du coût de l'énergie.

Tout au long du document, on note f les fonctions et $f(t)$ leur évaluation en t. Par exemple, il faut comprendre les paramètres $(\theta(t), \omega(t), E(t))$ du coût comme 3 scalaires, représentant la condition initiale tandis que son paramètre P est une fonction du temps.

Dans ce coût, le premier terme est le coût social pour l'agent qui mesure à quel point il a une consommation différente de sa consommation cible $\hat{E}$.

Le second terme est le coût de l'énergie sur le réseau. Lorsque p est grand, l'énergie est "chère" et l'agent a intérêt à produire de l'énergie ou à en consommer moins. On veut donc que Cout$[P]$ soit décroissante de P losrque p est positif. D'où le signe moins devant ce terme.

Le dernier terme correespond au terme "demand charge" proposé par Alasseur en partie 2.2 de son article [2]. En partie 4, elle propose de définir ce terme comme L_T^γ : $(q, \alpha) \mapsto \frac{K^\gamma}{2}(q-\alpha)^2$ dans son système de notation qui revient à introduire le terme $\frac{1}{2}\mu(P)^2$ avec nos notations.

On introduit enfin $u(\theta(t), \omega(t), E(t);\ t)$ le minimum du coût à l'instant t. Il faut comprendre qu'à l'instant t, $P(\tau)$ a déjà été fixé $\forall \tau \in [0, t[$. La fonction u est donc un minimum sur P restreint à l'intervalle $[t, T]$. On note ainsi $P\big|_{[t,T]}$ le paramètre de contrôle à l'instant t :

$$u(\theta(t), \omega(t), E(t);\ t) = \inf_{P\big|_{[t,T]}} \text{Cout}[P](\theta(t), \omega(t), E(t);\ t) \tag{35}$$

4.1.2 Hamilton-Jacobi-Bellman (HJB)

$$\begin{aligned} u(\theta(t), \omega(t), E(t);\ t) &= \inf_P \mathbb{E} \int_t^T \left[\eta \left(E(\tau) - \hat{E}(\tau) \right)^2 - P(\tau) \cdot p(\omega(\tau)) + \frac{1}{2}\mu (P(\tau))^2 \right] d\tau \\ &= \inf_P \mathbb{E} \left[\int_t^{t+dt} [..] + \int_{t+dt}^T [..] \right] \\ &= \inf_P \mathbb{E} \left[\left\{ \eta \left(E(t) - \hat{E}(t) \right)^2 - P(t) \cdot p(\omega(t)) + \frac{1}{2}\mu (P(t))^2 \right\} dt + u(\vec{x} + dX; t + dt) \right] \end{aligned} \tag{36}$$

Où l'on note $(\vec{x} + dX;\ t + dt) = (\theta(t) + d\theta, \quad \omega(t) + d\omega, \quad E(t) + dE;\ t + dt)$. En utilisant le lemme d'Ito sur u, on a :

$$u(\vec{x} + dX, t + dt) \quad = u(\vec{x}, t) + \frac{\partial u}{\partial t} dt + \Sigma_{i=1}^3 \frac{\partial u}{\partial x_i} dx_i + \frac{1}{2} \sum_{i,j=1}^3 \frac{\partial^2 u}{\partial x_i \partial x_j} \langle dx_i | \, | dx_j \rangle$$

Où la plupart des termes sont d'éspérance nulle. On a par exemple : $\mathbb{E}[dE] = \mathbb{E}[Pdt] + \mathbb{E}[\sigma dW^2] = Pdt$. D'où :

$$\begin{aligned} \mathbb{E}[u(\vec{x} + dX, t + dt)] = u(\vec{x}, t) + \Bigg[& \frac{\partial u}{\partial t} + \omega \frac{\partial u}{\partial \theta} + \frac{1}{m} \left(-\kappa\omega + P + \lambda \cdot r \sin(\Psi - \theta) \right) \frac{\partial u}{\partial \omega} + P \frac{\partial u}{\partial E} \\ & + \frac{\sigma^2}{2} \frac{\partial^2 u}{\partial E^2} + 2 \left(\frac{D}{m} \right)^2 \frac{\partial^2 u}{\partial \omega^2} \Bigg] dt \end{aligned} \tag{37}$$

Ensuite, on injecte (37) dans (36), les termes d'ordre 0 en dt s'annulent et on obtient à l'ordre 1 en dt :

$$\begin{aligned} 0 = \inf_{P(t)} \left[\frac{1}{2}\mu \left(P(t)\right)^2 + \left(-p(\omega(t)) + \frac{1}{m}\frac{\partial u}{\partial \omega} + \frac{\partial u}{\partial E} \right) P(t) \right] \\ + \eta \left(E(t) - \hat{E}(t) \right)^2 + \frac{\partial u}{\partial t} + \omega \frac{\partial u}{\partial \theta} + \frac{1}{m}\left(-\kappa\omega + \lambda \cdot r \sin(\Psi - \theta) \right) \frac{\partial u}{\partial \omega} \\ + \frac{\sigma^2}{2}\frac{\partial^2 u}{\partial E^2} + 2\left(\frac{D}{m}\right)^2 \frac{\partial^2 u}{\partial \omega^2} \end{aligned} \tag{38}$$

Où l'inf est sur un polynome d'ordre 2 en $P(t)$. On a donc directement :

$$P^*(t) = \frac{p(\omega(t)) - \frac{1}{m}\frac{\partial u}{\partial \omega} - \frac{\partial u}{\partial E}}{\mu} \tag{39}$$

On peut alors réinjecter $P^*(t)$ dans (38) pour obtenir l'équation de **Hamilton-Jacobi-Bellman (HJB)** :

$$\begin{aligned} 0 = \quad \frac{\left(p(\omega(t)) - \frac{1}{m}\frac{\partial u}{\partial \omega} - \frac{\partial u}{\partial E} \right)^2}{2\mu} + \eta \left(E(t) - \hat{E}(t) \right)^2 + \frac{\partial u}{\partial t} + \omega \frac{\partial u}{\partial \theta} + \frac{1}{m}\Big(-\kappa\omega + \lambda \cdot r \sin(\Psi - \theta) \Big) \frac{\partial u}{\partial \omega} \\ + \frac{\sigma^2}{2}\frac{\partial^2 u}{\partial E^2} + 2\left(\frac{D}{m}\right)^2 \frac{\partial^2 u}{\partial \omega^2} \end{aligned} \tag{40}$$

4.1.3 Forward Kolmogorov (Fokker-Planck)

Notre état est (θ, ω, E), et son évolution est régie par (1). D'où :

$$a = \begin{pmatrix} \omega \\ \frac{1}{m}(-\kappa\omega + P + \lambda \cdot r \sin(\Psi - \theta)) \\ P \end{pmatrix}, \quad b = \begin{pmatrix} 0 & 0 \\ \frac{2D}{m} & 0 \\ 0 & \sigma \end{pmatrix},$$

$$\text{et} \quad D = \frac{1}{2} b b^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{2D^2}{m^2} & 0 \\ 0 & 0 & \frac{\sigma^2}{2} \end{pmatrix}$$

En appliquant la formule proposée en annexe A, on obtient l'équation de **Fokker-Planck** associée :

$$\begin{aligned} \frac{\partial \rho_t}{\partial t} = -\frac{\partial}{\partial \theta}\left(\omega \rho_t\right) - \frac{1}{m}\frac{\partial}{\partial \omega}\left((-\kappa\omega + P + \lambda \cdot r \sin(\Psi - \theta))\rho_t \right) - \frac{\partial}{\partial E}\left(P \rho_t\right) \\ + 2\left(\frac{D}{m}\right)^2 \frac{\partial^2}{\partial \omega^2}\left(\rho_t\right) + \frac{\sigma^2}{2}\frac{\partial^2}{\partial E^2}\left(\rho_t\right) \end{aligned} \tag{41}$$

4.2 Recherche de l'équilibre de Nash

L'équation de HJB (40) décrit l'évolution de u tandis que l'équation de Fokker-Planck (11) décrit l'évolution de ρ_t. Elles sont liées par la relation suivante :

$$r(t)\exp\left(i\Psi(t)\right) = \int \rho_t((\theta, \omega, P))\exp\left(i\theta\right) d\theta d\omega dP \tag{42}$$

HJB évolue de l'instant final vers l'instant initial, alors que Fokker-PLanck évolue de l'instant initial vers l'instant final. En conséquence, on ne peut pas les résoudre ensemble. Pour espérer trouver un équilibre de Nash, on résout Focker-Planck (11) une première fois en utilisant une *stratégie initiale* P^0. On obtient ainsi un r et un Ψ. On peut alors faire plusieurs itérations de :

- injecter r et Ψ dans HJB (40) pour obtenir un nouveau u et une nouvelle stratégie P.
- injecter P dans Fokker-Planck (41) pour obtenir une nouvelle ρ_t.
- utiliser ce ρ_t pour calculer un nouveau r et une nouvelle Ψ.

On continue ces itérations jusqu'à convergence, c'est-à-dire jusqu'à ce que P et ρ_t ne changent plus d'une itération à l'autre. Cette solution devrait permettre de trouver l'équilibre de Nash mais elle comprend l'évolution d'une densité à 3 dimensions sur un temps $T = 24$h très long devant les temps caractéristiques de son équation différentielle. Ce problème est donc très lourd numériquement. Après avoir codé la résolution de Fokker-Planck qui est présentée dans la sous-section suivante, j'ai passé mes derniers jours de stage à chercher à simplifier le problème en considérant l'état (θ, ω) plutôt que (θ, ω, E). J'ai conclu mon stage en expliquant analytiquement que cette évolution ne se réduisait pas à un état 2D comme on voit en annexe B.2

4.3 Implémentation numérique de Fokker-Planck en 2D

On détaille ci-dessous une implémentation de la dynamique de $\rho_t(\theta, \omega, E)$ dans le cas particulier où la stratégie $P_t(\theta, \omega, \cancel{E})$ est homogène dans la direction E. Ainsi, on a $\rho_t(\theta, \omega, \cancel{E})$ On se ramène ainsi à l'équation différentielle suivante :

$$\frac{\partial \rho_t}{\partial t} = -\frac{\partial}{\partial \theta}(\omega \rho_t) - \frac{1}{m}\frac{\partial}{\partial \omega}\left((-\kappa\omega + P + \lambda \cdot r\sin(\Psi - \theta))\rho_t\right) + 2\left(\frac{D}{m}\right)^2 \frac{\partial^2}{\partial \omega^2}(\rho_t) \quad (43)$$

Assez classiquement, on discrétise la densité en un array de taille (N_theta, N_omega) tel que $\sum_{i,j} \rho_t(i,j) = 1$ et chaque coefficient est compris entre 0 et 1. Le membre de droite est l'opérateur évolution appliqué à cette densité. Parmi les 3 termes du membre de droite, le premier agit dans la direction theta de manière explicite et très rapide. Les deux autres termes agissent dans la direction de nu de manière plus complexe en changeant la géométrie du nuage de densité.

Ce problème n'est pas simple car le premier terme évolue sur des ordres de grandeur très différents des deux autres. Pour un pas de temps donné, le premier terme provoque un déplacement de la masse sur une distance de l'ordre de π (la direction θ est 2π périodique) tandis que la direction ω se réarrange sur des distances de l'ordre du pixel. Cet opérateur particulier se prête parfaitement à un splitting de Strang [6]. Pour passer d'un pas de temps au suivant, on réalise les étapes suivantes :

- On calcule r et Ψ de l'instant t.
- On réalise explicitement l'évolution due au terme $-\frac{\partial}{\partial \theta}(\omega \rho_t)$ durant $\frac{\mathrm{d}t}{2}$.
- On applique Runge-kuta d'ordre 4 pour l'opérateur suivant nu pour une évolution $\mathrm{d}t$.
- On réalise explicitement l'évolution due au terme $-\frac{\partial}{\partial \theta}(\omega \rho_t)$ durant $\frac{\mathrm{d}t}{2}$.

En utilisant ce code, on obtient essentiellement 2 types de comportements distincts en fonction du paramètre λ du modèle :

- Lorsque λ est faible ou nul, les agents ne se synchronisent pas et se répartissent quasiment homogènement suivant theta. La dynamique est alors régie par le secon terme de l'opérateur et la densité se place globalement autour d'une vitesse angulaire moyenne ω_{eq}.
- Lorsque $\lambda \geq$ N_theta $\cdot$ N_omega, les agents arrivent à se synchroniser et alors on observe une masse regroupée autour d'une vitesse angulaire ω_{eq} et d'un angle. Durant $\mathrm{d}t$, on observe que l'angle moyen de cette masse se déphase de $\omega_{eq} \cdot \mathrm{d}t$.

Dans le premier cas, on se retrouve avec une dynamique à une dimension régie par un terme de drift affine $-\kappa\omega + P$, et un terme de diffusion $2\left(\frac{D}{m}\right)^2 \frac{\partial^2}{\partial \omega^2}(\rho_t)$. On observe ainsi une gaussienne dont le centre est déterminé par le premier terme via $\omega_{eq} = \frac{P}{\kappa}$; et dont la largeur croît avec $\frac{D}{m}$. On peut observer en figure 9 la formation de l'état stationnaire

à partir d'une population relativement en phase. En figure 10, on voit comment cet état stationnaire dépend de la valeur de $\frac{D}{m}$. On vérifie bien que la densité est centrée en $\omega_{eq} = \frac{P}{\kappa}$.

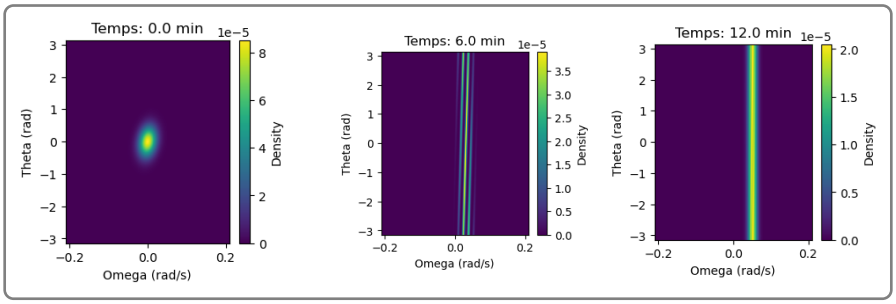


Figure 9 – Evolution temporelle de la densité d'état dans l'espace 2D pour **M = 1e3, P0 = 1e-1, lambda = 1, kappa = 1e0, D = 1e-1**. On voit ici comment se forme la densité d'état homogène en théta lorsque λ est trop faible.*(Code en annexe C.3)*

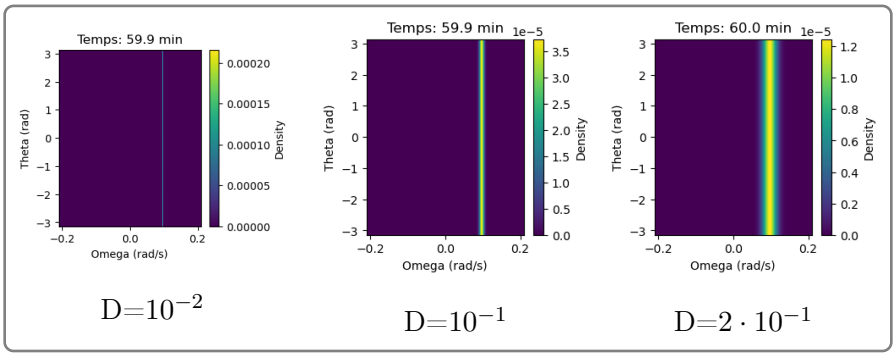


Figure 10 – État stationnaire atteint au bout de plusieurs minutes d'évolution pour différentes valeurs de D et pour les paramètres suivant : **M = 1e3, P0 = 1e-1, lambda = 1, kappa = 1e0**. On voit ici comment le terme de bruit D iflue la dynamique de la densité d'état. *(Code en annexe C.3)*

Lorsque l'on augmente λ, on passe une transition de phase au-delà de laquelle l'état des agents se synchronise. On obtient ainsi à chaque instant une densité regroupée autour d'un $\langle\theta\rangle$ et d'un $\langle\omega\rangle$. On peut voir la formation de cet état synchronisé sur la figure ci-dessous :

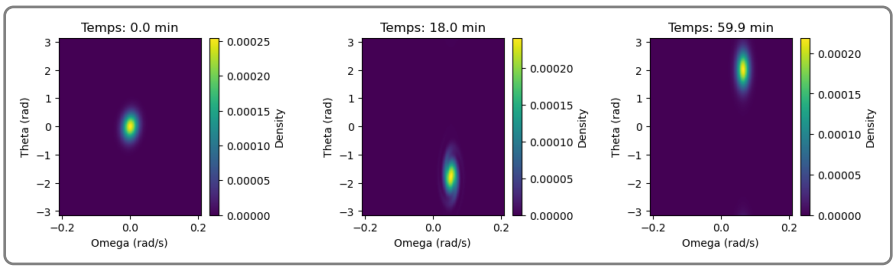


Figure 11 – Evolution temporelle de la densité d'état dans l'espace 2D pour **M = 1e3, P0 = 1e-1, lambda = 10e6, kappa = 1e0, D = 1e-1**. On voit ici comment se forme la densité d'état homogène en théta lorsque λ est suffisamment élevé pour avoir un état synchronisé.*(Code en annexe C.3)*

Sur la première image, on voit un état initial quelconque. Lors de l'évolution de cet état, le terme en $\frac{\lambda}{m}\frac{\partial}{\partial\omega}\left(r\sin(\Psi-\theta)\rho_t\right)$ joue un rôle différent sur les agents en fonction de théta :

- Il ralentit les agents en avance sur le groupe. En terme de densité, il déplace vers la gauche la masse la plus haute sur la figure 10.
- Il accélère les agents en retard. En terme de densité, il déplace vers la droite la masse la plus basse sur la figure 10.

Ces phénomènes conduisent à la formation d'une spirale pendant le régime transitoire comme on peut voir sur la vignette du milieu de la figure 10. Lorsque le temps s'écoule, cette spirale s'enroule et devient une masse comme on peut le voir sur la $3^{\text{ème}}$ vignette qui correspond à un état stationnaire.

Conclusion

Au cours de ce stage, nous avons développé un cadre de modélisation par jeux à champ moyen pour étudier l'équilibrage de la production et de la consommation électriques intermittentes. Après avoir formalisé les équations en l'absence de bruit et validé numériquement ce cas de référence, nous avons étendu le modèle pour inclure la composante stochastique pour tenir compte de la variabilité inhérente aux sources renouvelables ; et un paramètre de contrôle plus riche pour mieux refléter le comportement des consommateurs. Ces ajouts ont rendu le problème très lourd numériquement et une réflexion s'est imposée pour vérifier que les équations ne soient pas simplifiables. Le stage s'est conclu par l'affirmation que la richesse du modèle imposait effectivement une grande puissance de calcul. Les prochaines étapes du projet sont ainsi à priori les suivantes :

- *Mise à l'échelle sur cluster HPC* : déploiement de l'implémentation sur un cluster pour trouver des équilibres de Nash pour ces densités d'état à 3 dimensions.
- *Populations hétérogènes* : introduction de sous-populations d'agents (différentes fonctions d'utilité, capacités de stockage, comportements adaptatifs) afin d'étudier l'émergence de dynamiques différenciées et leurs effets sur la stabilité globale.
- *Topologies réalistes* : substitution du graphe complet par des réseaux à connectivité limitée (petits-mondes, réseaux en grille, topologies issues de données réelles) pour évaluer l'impact de la structure du réseau sur la synchronisation et la robustesse.
- *Game design pour l'autorité centrale* : élaboration de mécanismes d'incitation et de schémas de tarification optimaux, afin de guider les agents vers un équilibre de Nash plus proche d'un équilibre sociétal.

Remerciements

Je voudrais remercier toute l'équipe pour son accueil et sa bonne humeur pendant ces quelques mois. Un grand merci à Denis Ullmo pour son encadrement, sa disponibilité et ses conseils toujours clairs. Merci aussi à Guillaume Roux pour toutes les discussions et l'aide qu'il m'a apportée, et à Loumi Gatouillat pour les heures très agréables que l'on a passées ensemble sur tableau noir.

Ce stage m'a vraiment permis de découvrir le quotidien du métier de chercheur – et de me rendre compte que ça me plaît. J'ai appris plein de choses dans une super ambiance.

Merci pour le temps que vous m'avez accordé, merci pour votre confiance, vos idées, vos discussions. Merci de m'avoir montré EDF. Merci.

ANNEXE

A Rappel de Fokker-Planck

Voici un rappel de la forme générale de l'équation de Fokker-Planck pour un processus stochastique. Un rappel plus rigoureux avec une preuve accessible pour des physiciens se trouve en partie (3.1) de l'article de Martin [3]. En tout cas, voici une version de l'équation dans notre formalisme : Notant X_t un processus stochastique de dimension N ($N = 3$ dans notre cas), on note $\rho_t(\vec{x})$ sa densité de probabilité à l'instant t. On note $a(x,t) \in \mathbb{R}^N$ le drift du processus et $b(x,t) \in \mathbb{R}^{N\times M}$ sa matrice de diffusion. On introduit aussi dW_t un processus de Wiener de dimension M. L'évolution de X_t est donc donnée par :

$$dX_t = a(X_t,t)dt + b(X_t,t)dW_t \tag{44}$$

L'équation de Fokker-Planck associée est donnée par :

$$\frac{\partial \rho_t}{\partial t} = -\Sigma_{i=1}^{N} \frac{\partial}{\partial x_i}(a\rho_t) + \Sigma_{i,j=1}^{N} \frac{1}{2}\frac{\partial^2}{\partial x_i \partial x_j}(D_{ij}\rho_t) \tag{45}$$

Où l'on a introduit $D = \frac{1}{2}bb^T$ la matrice de diffusion du processus.

B Modèle sans bruit

On essaie dans cette section de trouver la solution explicite du problème d'optimisation. Cette méthode aboutit à des solutions explicites pour un état à deux dimensions (θ, ω). On voit dans la première sous-partie que lorsque le problème est rédigé à 2 dimensions, les équations résultantes du mouvement ont un ordre plus élevé pour englober la richesse du problème.

B.1 Méthode de Lagrange

On définit dans cette partie le coût $C[\theta, \omega, E, P]$ sur un journée. On lui ajoute ensuite les contraintes qui font sens avec des multiplicateurs de Lagrange pour faire une minimisation sous contrainte :

$$C[\theta,\omega,E,P] = \int_0^T \left[\frac{1}{2}\eta\Big(E(\tau) - \hat{E}(\tau)\Big)^2 - P(\tau)\cdot p\Big(\omega(\tau)\Big) + \frac{1}{2}\mu\Big(P(\tau)\Big)^2\right] \mathrm{d}t \tag{46}$$

On y ajoute les contraintes locales suivantes :

$$\begin{aligned} &\dot{\theta} = \omega && \text{Dont les multiplicateurs de Lagrange sont :} && \alpha_t \\ &\dot{\omega} = \frac{1}{m}\Big(-\kappa\omega + P + \lambda r\sin(\Psi - \theta)\Big) && && \beta_t \\ &\dot{E} = P && && \gamma_t \end{aligned}$$

On ajoute aussi la contrainte globale :

$$E(T) = \hat{E}(T) \qquad \text{Dont le multiplicateur de Lagrange (scalaire) est :} \quad \delta$$

Le coût se réécrit ainsi :

$$\tilde{C}\left[\theta,\omega,E,P,\alpha,\beta,\gamma\right](\delta) = \int_0^T \Bigg[\frac{1}{2}\eta\left(E-\hat{E}\right)^2 - P\cdot p\left(\omega\right) + \frac{1}{2}\mu P^2 + \alpha_t(\dot{\theta}-\omega) + \beta_t\Bigg(\dot{\omega} - \frac{1}{m}\Big(-\kappa\omega + P + \lambda r\sin(\Psi-\theta)\Big)\Bigg) + \gamma_t(\dot{E}-P)\Bigg]\mathrm{d}t + \delta\Big(E(T)-\hat{E}(T)\Big) \tag{47}$$

Avec ces contraintes, on considère tous les paramètres du coût comme indépendants et on peut réaliser une IPP sur certains termes pour expliciter la dépendance du coût en les paramètres ω et θ :

$$\tilde{C}\left[\theta,\omega,E,P,\alpha,\beta,\gamma\right](\delta) = \int_0^T \Bigg[\frac{1}{2}\eta\left(E-\hat{E}\right)^2 - P\cdot p\left(\omega\right) + \frac{1}{2}\mu P^2 - \dot{\alpha_\tau}\theta - \alpha_\tau\omega - \dot{\beta_\tau}\omega - \beta_\tau\frac{1}{m}\Big(-\kappa\omega + P + \lambda r\sin(\Psi-\theta)\Big) - \dot{\gamma_\tau}E - \gamma_\tau P\Bigg]\mathrm{d}\tau + \delta\Big(E(T)-\hat{E}(T)\Big) + \Big[\beta_t\omega + \gamma_t E\Big]_0^T \tag{48}$$

Ce coût ayant 7 paramètres, l'optimisation donne à priori 7 équations :

$$\frac{\delta\tilde{C}}{\delta\theta} = 0 \quad \Rightarrow \quad \dot{\alpha_t} - \frac{\beta_t}{m}\lambda r\cos(\Psi-\theta) = 0 \tag{49}$$

$$\frac{\delta\tilde{C}}{\delta\omega} = 0 \quad \Rightarrow \quad P(t)\cdot p'\Big(\omega(t)\Big) + \alpha_t + \dot{\beta_t} + \frac{\beta_t}{m}\kappa = 0 \tag{50}$$

$$\frac{\delta\tilde{C}}{\delta P} = 0 \quad \Rightarrow \quad -p\Big(\omega(t)\Big) + \mu P(t) - \frac{\beta_t}{m} - \gamma_t = 0 \tag{51}$$

$$\frac{\delta\tilde{C}}{\delta E} = 0 \quad \Rightarrow \quad \eta\left(E-\hat{E}\right) - \dot{\gamma_t} = 0 \tag{52}$$

$$\frac{\delta\tilde{C}}{\delta\alpha} = 0 \quad \Rightarrow \quad \dot{\theta} = \omega \tag{53}$$

$$\frac{\delta\tilde{C}}{\delta\beta} = 0 \quad \Rightarrow \quad \dot{\omega} = \frac{1}{m}\Big(-\kappa\omega + P + \lambda r\sin(\Psi-\theta)\Big) \tag{54}$$

$$\frac{\delta\tilde{C}}{\delta\gamma} = 0 \quad \Rightarrow \quad \dot{E} = P \tag{55}$$

$$\frac{\partial\tilde{C}}{\partial\delta} = 0 \quad \Rightarrow \quad E(T) = \hat{E}(T) \tag{56}$$

$$\frac{\partial\tilde{C}}{\partial\omega(T)},\ \frac{\partial\tilde{C}}{\partial\beta_T},\ \frac{\partial\tilde{C}}{\partial\gamma_T},\ \frac{\partial\tilde{C}}{\partial E(T)} = 0 \quad \Rightarrow \quad \beta_T = 0,\ \omega(T) = 0,\ E(T) = 0,\ \delta+\gamma_T = 0 \tag{57}$$

On peut réécrire ce système d'équations pour faire disparaître les fonctions $(\alpha_t,\ \beta_t,\ \gamma_t)$ du système. On utilise la contrainte globale en E :

$$0 = E(T) - \hat{E}(T) = \int_0^T (P-\hat{P}) = E(t) - \hat{E}(t) + \int_t^T (P-\hat{P})$$

Pour faire disparaître formellement la variable E des équations. On comprend qu'en réalité, la dynamique dépend de cette variable même si on sait la "cacher" dans $\int_t^T (P-\hat{P})$. On obtient ainsi le jeu de 3 équations différentielles suivant en (θ,ω,P) :

$$
\begin{aligned}
\dot{\theta} &= \omega \\
\dot{\omega} &= \frac{1}{m}\Big(-\kappa\omega + P + \lambda r \sin(\Psi - \theta)\Big) \\
0 &= m\Big(\mu\ddot{P} - p''\ddot{\omega} - \eta(P - \hat{P})\Big) + \kappa\left(\mu\dot{P} - p'\dot{\omega} + \eta\int_t^T (P - \hat{P})\right) \\
&\quad + \lambda r\cos(\Psi - \theta)\left(\mu(P(t) - P(T)) - \Big(p - p(\omega(T))\Big)\right) + \eta\int_t^T \mathrm{d}\tau\left[\int_\tau^T \Big(P - \hat{P}\Big)\right] \\
&\quad + \dot{P}p' + Pp''\dot{\omega}
\end{aligned}
\tag{58}
$$

B.2 HJB

Lorsque le bruit est nul, on sait retrouver exactement les équations de la partie précédente en faisant HJB avec un bruit nul et en utilisant $E = \int P$ pour faire disparaître la variable E. Le problème essentiel est que lorsque l'on ajoute le bruit à E, cette égalité n'est plus vérifiée et on est obligés de considérer E comme une variable d'état à part entière même si on connaît le paramètre de contrôle P.

Cela justifie qu'un code complet avec HJB doit comporter la simulation de Fokker-Plank à 3 dimensions durant 24h pour chaque itération de la recherche de l'équilibre de Nash. Cet algorithme de recherche d'équilibre demande donc nécessairement une grande puissance de calcul.

C Codes principaux

C.1 Code en Julia pour les figures 2 à 4

```
using DifferentialEquations, Plots, LinearAlgebra, Distributions, ProgressMeter, NPZ, LaTeXStrings, Measures
gr()

# --- Calcul de l'energie potentielle ---
Theta = range(-π, 3π, length=1000)
K = 1.0
P_liste = [0.1, 0.4, 0.6, 1.1]
E_val = Array{Float64}(undef, length(Theta), length(P_liste))
Theta_max = zeros(length(P_liste))
E_max = zeros(length(P_liste))
for (j, P) in enumerate(P_liste)
    function E(θ)
        return - 2* P * θ - 2* K * cos(θ)
    end
    E_val[:,j] = E.(Theta)
    if P < K
        Theta_max[j] = π - asin(P/K)
        E_max[j] = E(Theta_max[j])
    end
end

# --- Trace de l'energie potentielle ---
p1 = plot(Theta, E_val[:,1], title=L"P = 0.1~[\mathrm{rad.s^{-2}}]", color= [0 < theta < 2π ? :blue : :grey for theta in Theta], linewidth=2, label="V(θ)")
p2 = plot(Theta, E_val[:,2], title=L"P = 0.4~[\mathrm{rad.s^{-2}}]", color= [0 < theta < 2π ? :blue : :grey for theta in Theta], linewidth=2, label="V(θ)")
p3 = plot(Theta, E_val[:,3], title=L"P = 0.6~[\mathrm{rad.s^{-2}}]", color= [0 < theta < 2π ? :blue : :grey for theta in Theta], linewidth=2, label="V(θ)")
p4 = plot(Theta, E_val[:,4], title=L"P = 1.1~[\mathrm{rad.s^{-2}}]", color= [0 < theta < 2π ? :blue : :grey for theta in Theta], linewidth=2, label="V(θ)")
p1 = scatter!(p1, [Theta_max[1]], [E_max[1]], color=:black, linestyle=:dash, label="Max local de V")
```

```
p2 = scatter!(p2, [Theta_max[2]], [E_max[2]], color=:black, linestyle=:dash, label="Max local de V")
p3 = scatter!(p3, [Theta_max[3]], [E_max[3]], color=:black, linestyle=:dash, label="Max local de V")
plt = plot(p1, p4, p3, p2,
    layout=(2,2),
    size=(1200,900),
    xlabel=L"\theta_i~[\mathrm{rad}]",
    ylabel=L"V(\theta_i)",
    xticks=([-π, 0, π, 2π, 3π], [L"-\pi", L"0", L"\pi", L"2\pi", L"3\pi"]),
    legend=:topright,
    xlabelfontsize=22,   # Taille de la legende de l'axe x
    ylabelfontsize=22,    # Taille de la legende de l'axe y
    titlefontsize=22,     # Taille du titre
    tickfontsize=18,      # Taille des ticks
    legendfontsize=12,     # Taille de la legende
    left_margin=10mm,
    bottom_margin=10mm
)
savefig(plt, "energie_potentielle.pdf")
display(plt)
```

Listing 1 – Code figure 2

```
using DifferentialEquations, Plots, LinearAlgebra, Distributions, ProgressMeter, LaTeXStrings, Measures
gr()

# Parametres
K = 1.0
α = 0.3
γ = 0.0
τ = 0.0
tspan = (0.0, 600.0)
ε = 2π * 1e-1
P_liste = [0.4, 0.6]
dephasage = 0.0

# --- Calcul des points bleus et verts ---
n = 20000
m = length(P_liste)
zeta_theta = π
zeta_omega = 30.0
dTheta = rand(Uniform(-zeta_theta, zeta_theta), n)
dOmega = rand(Uniform(-zeta_omega, zeta_omega), n)

results = Array{Float64, 3}(undef, n, m, 2)
colors = Matrix{Symbol}(undef, n, m)

@showprogress for i in 1:n
    u0 = [dTheta[i], dOmega[i]]
    for (j, P) in enumerate(P_liste)
        p = (P, α, K, γ, τ)
        if τ == 0.0
            function f_ode(du, u, p, t)
                P, α, K, γ, τ = p
                θ, ω = u
                du[1] = ω
                du[2] = 2P - (α + γ) * ω - 2K * sin(θ + dephasage)
            end
            prob = ODEProblem(f_ode, u0, tspan, p)
            sol = solve(prob, Tsit5(), reltol=1e-8, abstol=1e-8, saveat=0.01)
        else
            function h(p, t)
                scale = 1.0 + 0.1 * tanh(t / 2)
                return [u0[1] * scale, u0[2] * scale]
            end
            function f_dde(du, u, h, p, t)
                P, α, K, γ, τ = p
                θ, ω = u
                ωτ = h(p, t - τ)[2]
                du[1] = ω
                du[2] = 2P - α * ω - 2K * sin(θ + dephasage) - γ * ωτ
            end
```

```
            prob = DDEProblem(f_dde, u0, h, tspan, p; constant_lags=[τ])
            sol = solve(prob, reltol=1e-8, abstol=1e-8, saveat=0.01)
        end
        dθi = sol.u[1][1]
        dωi, dωf = sol.u[1][2], sol.u[end][2]
        results[i, j, :] = [dθi, dωi]
        colors[i,j] = abs(dωf) < ε ? :green : :blue

    end
end

# --- Calcul du cycle attracteur ---
P_liste = [0.4, 0.6]
trajectoire = Array{Float64, 3}(undef, 2, 2, 300)
u0= [0.0, 15.0]
tspan = (0.0, 600.0)
for (j, P) in enumerate(P_liste)
    p = (P, α, K, γ, τ)
    function f_ode(du, u, p, t)
        P, α, K, γ, τ = p
        θ, ω = u
        du[1] = ω
        du[2] = 2P - (α + γ) * ω - 2K * sin(θ + dephasage)
    end
    prob = ODEProblem(f_ode, u0, tspan, p)
    sol = solve(prob, Tsit5(), reltol=1e-8, abstol=1e-8, saveat=0.01)
    θs = [u[1] for u in sol.u[end-299:end]]
    ωs = [u[2] for u in sol.u[end-299:end]]
    trajectoire[j, 1, :] = θs
    trajectoire[j, 2, :] = ωs
end

function wrap_to_pi(arr)
    return [mod(x + π, 2π) - π for x in arr]
end
trajectoire[:,1,:] = wrap_to_pi(trajectoire[:,1,:])

for i in axes(trajectoire, 1)
    for j in axes(trajectoire, 2)
        # Recupere les indices de tri selon la valeur de la 2e coordonnee
        idx = sortperm(trajectoire[i, 1, :])
        # Trie les deux coordonnees selon ces indices
        trajectoire[i, 1, :] = trajectoire[i, 1, idx]
        trajectoire[i, 2, :] = trajectoire[i, 2, idx]
    end
end

# --- affichage des points bleus et verts ---
p1 = scatter(results[:,1,1], results[:,1,2], title=L"P = 0.4~[\mathrm{rad.s^{-2}}]", color=colors[:,1], markersize = 2, markerstrokecolor = colors[:,1], legend=false,label = "")
p2 = scatter(results[:,2,1], results[:,2,2], title=L"P = 0.6~[\mathrm{rad.s^{-2}}]", color=colors[:,2], markersize = 2, markerstrokecolor = colors[:,2], legend=false, label = "")
scatter!(p1, [NaN], [NaN], color=:green, markerstrokecolor=:green, label="Converge vers le point fixe")
scatter!(p1, [NaN], [NaN], color=:blue, markerstrokecolor=:blue, label="Atteint le cycle limite")

# --- affichage du cycle attracteur ---
plot!(p1, trajectoire[1, 1, :], trajectoire[1, 2, :], color=:red, linewidth=3, label="Attracteur")
plot!(p2, trajectoire[2, 1, :], trajectoire[2, 2, :], color=:red, linewidth=3, label="")

# --- affichage du point fixe ---
scatter!(p1, [asin(P_liste[1]/K)], [0], color=:red, markerstrokecolor=:red, marker=:star6, markersize=12, label="Point fixe stable")
scatter!(p2, [asin(P_liste[2]/K)], [0], color=:red, markerstrokecolor=:red, marker=:star6, markersize=12, label="")

# --- generation et enregistrement de l'image ---
plt = plot(p1, p2,
    layout=(1,2),
```

```
    size=(1200,600),
    legend=:topleft,
    xlabel=L"\theta_i~[\mathrm{rad}]",
    ylabel=L"\omega_i~[\mathrm{rad.{s}^{-1}}]",
    xlabelfontsize=22,
    ylabelfontsize=22,
    titlefontsize=22,
    tickfontsize=18,
    legendfontsize=12,
    left_margin=8mm,
    bottom_margin=15mm,
    top_margin=10mm,
    xticks=([-π,-π/2, 0, π/2, π], [L"-\pi", L"-\frac{\pi}{2}", L"0", L"\frac{\pi}{2}", L"\pi"]),
    yticks=([-30, 0, 30], [L"-30", L"0", L"30"]),
    framestyle=:box,
    framestyle_color=:black,
    framestyle_width=1.5,
    grid=:on,
)
display(plt)
savefig(plt, "diagramme_bifurcation.pdf")
```

Listing 2 – Code Figure 3

```
using DifferentialEquations, Plots, LinearAlgebra, Distributions, ProgressMeter, NPZ, LaTeXStrings, Measures

# Parametres
K = 8.0
tspan = (0.0, 1000.0)
ε = 1e-3
Omega0 = 1e-1

n = 100
Alpha = range(1e-1, 6, length=n)
P_liste = range(1e-1, 10, length=n)
m = length(P_liste)

results = Array{Float64, 3}(undef, n, m, 2)
colors = Matrix{Symbol}(undef, n, m)

@showprogress for i in 1:n
    α = Alpha[i]
    for (j, P) in enumerate(P_liste)
        if P>K
            colors[i,j] = :grey
        else
            u0 = [π - asin(P / K), Omega0]
            p = (P, α, K)
            function f_ode(du, u, p, t)
                P, α, K = p
                θ, ω = u
                du[1] = ω
                du[2] = 2P - α * ω - 2K * sin(θ)
            end
            prob = ODEProblem(f_ode, u0, tspan, p)
            sol = solve(prob, Tsit5(), reltol=1e-8, abstol=1e-8)
            colors[i,j] = abs( sol.u[end][2] / (2π) ) < ε ? :red : :yellow
        end
        results[i, j, 1] = α
        results[i, j, 2] = P
    end
end

# --- Affichage ---
gr()
s = scatter(
    results[:,:,1],
    results[:,:,2],
    color=colors[:,:],
    markersize = 2,
    markerstrokecolor = colors[:,:],
    legend=false,
```

```
    label = ""
    )
scatter!(s, [NaN], [NaN], color=:red, markerstrokecolor=:red, label="Point fixe globalement stable")
scatter!(s, [NaN], [NaN], color=:grey, markerstrokecolor=:grey, label="Cycle attracteur globalement stable")
scatter!(s, [NaN], [NaN], color=:yellow, markerstrokecolor=:yellow, label="Point fixe et cycle attracteur localement stables")
plt = plot(s,
    size=(900,600),
    legend = :bottomright,
    title=L"K = 8.0~[\mathrm{rad.s^{-2}}]",
    xlabel=L"\alpha~\textit{(dumping)}~[\mathrm{s^{-1}}]",
    ylabel=L"P~[\mathrm{rad.s^{-2}}]",
    xlabelfontsize=22,
    ylabelfontsize=22,
    titlefontsize=22,
    tickfontsize=18,
    legendfontsize=12,
    left_margin=8mm,
    bottom_margin=5mm,
    top_margin=5mm,
    )
savefig(plt, "diagramme_de_bifurcation_par.pdf")
display(plt)
```

Listing 3 – Code Figure 4

C.2 Code en Python pour la Sous-section (3.2)

Dans ce code, on utilise @njit pour compiler (njit = *numba just in time*). L'option paralel = True permet de paralléliser les boucle for en remplaçant range par prange.

```
M = 1e3
D = M/5
eps = 1
lam = 1

import numpy as np
import scipy.integrate as integrate
from tqdm import tqdm
import matplotlib.pyplot as plt
import scipy.optimize as optimize
from numba import njit
from numba import prange
import cvxpy as cp

# --- Parametres du modele ---
T = 24*3600   # Duree totale de la simulation (24 heures en secondes)
n = 10  # Nombre de pas de temps par douche
Dt = 15 * 60  # Pas de temps pour la simulation (15 minutes en secondes)
N = int(n* 24 * 3600 / Dt)  # Nombre de groupes d'agents (toutes les 15 minutes sur 24 heures)
dT = T/N
t_vals = np.linspace(0, T, N)
n_points = 10 # Nombre de points pour les integrations numeriques

# --- Creation de la densite de probabilite des douches (rho_array) ---
def double_gaussienne(t, mu1=8*3600, mu2=20*3600, sigma=3600):
    """Double gaussienne centree a 8h et 20h pour modeliser les pics de douches."""
    return 0.5 * (np.exp(-0.5 * ((t - mu1) / sigma) ** 2) + np.exp(-0.5 * ((t - mu2) / sigma) ** 2))
rho_vals = double_gaussienne(t_vals)  # Calcul de la densite de probabilite pour chaque groupe d'agents
rho_array = rho_vals / np.sum(rho_vals)  # Normalisation pour que l'integrale soit 1

# --- Fonction de puissance moyenne ---
@njit
def P(t, puissance_moyenne=1., ecart_type=3 * 3600):
    """Puissance moyenne consommee par les agents a l'instant t."""
```

```
    return puissance_moyenne * (np.exp(-0.5 * ((t - T/2) / ecart_type) ** 2) )

# --- Calcul de C0 ---
def C_sans_renormalisation(t, f_lgn, Dt, dT, N):
    return np.sum( f_lgn* np.exp(- ((t - np.arange(N)*dT) / Dt)**2) )
integrale_P = integrate.quad(lambda t: P(t), 0, T)[0]  # Integrale totale de la puissance sur la journee
integrale_C_sans_renormalisation = integrate.quad(lambda t: C_sans_renormalisation(t, rho_array, Dt, dT, N), 0, T)[0]
C0 = integrale_P / integrale_C_sans_renormalisation

x_100 = optimize.root_scalar(lambda x: np.sinh(x) - 100, bracket=[0, 10]).root
omega0 = 2*np.pi*0.05/ x_100  # Frequence a ne pas depasser

print("Pas de discretisation :", dT, "s")
print("temps caracteristique du reseau M/D :", round(M/D, 2), "s")
print("omega0 :", round(omega0, 2), "rad/s")

# --- Fonctions principales du modele ---
@njit
def C_i(t, i, C0):
    return C0 * np.exp(- ((t - i*dT) / Dt)**2)

@njit
def C_moyen(t, f_lgn, C0, Dt, dT, N):
    return C0 * np.sum( f_lgn* np.exp(- ((t - np.arange(N)*dT) / Dt)**2) )

@njit
def omega_trapeze(t1,t2, f_lgn, C0, Dt, dT, N, n_points, D, M):
    t_grid = np.linspace(t1, t2, n_points)
    integrand = np.empty_like(t_grid)
    for idx in range(n_points):
        tt = t_grid[idx]
        integrand[idx] = np.exp(D/M*(tt-t2)) * (P(tt) - C_moyen(tt, f_lgn, C0, Dt, dT, N))
    val_integrale = np.trapz(integrand, t_grid)
    return val_integrale / M

@njit
def omega_vect(f_lgn, C0, Dt, dT, N, n_points, D, M):
    res = np.zeros(N)
    omega_intermediaire = np.empty(N)
    for i in range(N):
        t1 = i * dT
        t2 = (i + 1) * dT
        omega_intermediaire[i] = omega_trapeze(t1, t2, f_lgn, C0, Dt, dT, N, n_points, D, M)
        for j in range(i):
            #print(np.exp(D/M * (j+1-i)*dT))
            res[i] += omega_intermediaire[j] * np.exp(D/M * (j+1-i)*dT)
    return res

@njit(parallel=True)
def circular_slice(arr, i_min, i_max):
    N = len(arr)
    start = i_min % N
    end = i_max % N
    if start <= end:
        return arr[start:end + 1]
    else:
        size = N - start + end + 1
        out = np.empty(size, arr.dtype)
        for k in prange(N - start):
            out[k] = arr[start + k]
        for k in prange(end + 1):
            out[N - start + k] = arr[k]
    return out

@njit(parallel=True)
def Couts_integrales(f_lgn, Dt, dT, N, n_points, D, M, eps, omega_array, omega0):
    res = np.empty(N)
    for i in prange(N):
        omega_slice = circular_slice(omega_array, i - 3*n, i + 3*n)
        t_slice = circular_slice(np.arange(0, N) * dT, i - 3*n, i + 3*n)
```

```
        integrand = np.empty_like(omega_slice)
        for j in range(len(omega_slice)):
            integrand[j] = np.sinh(omega_slice[j]/omega0) * C_i(t_slice[j], i, C0)
                #np.sinh(omega_slice[j]/omega0)
        res[i] = eps * np.trapz(integrand, np.arange(0, 6*n+1 )*dT)
    return res

@njit(parallel=True)
def Couts_totaux(j, Couts_integral):
    res = np.empty(N)
    for i in prange(N):
        res[i] = lam * (i - j) ** 2 - Couts_integral[i]
    return res

def affichage(f_lgn, omega_array, commentaire = ""):
    fig, axs = plt.subplots(1, 2, figsize=(14, 5))
    axs[0].plot(t_vals / 3600, [C_moyen(t, f_lgn, C0, Dt, dT, N) for t in t_vals],
         label="Consommation d'energie")
    axs[0].plot(t_vals / 3600, [C_moyen(t, rho_array, C0, Dt, dT, N) for t in
        t_vals], label="Consommation sans opti")
    axs[0].plot(t_vals / 3600, [P(t) for t in t_vals], label='Puissance moyenne')
    axs[0].set_xlabel('Heure de la journee', fontsize=15)
    axs[0].set_ylabel('Puissance (W)', fontsize=15)
    axs[0].set_title('Consommation et puissance moyennes {}'.format(commentaire),
        fontsize=15)
    axs[0].legend()
    axs[1].plot(t_vals / 3600, omega_array, label="Omega")
    axs[1].set_xlabel('Heure de la journee', fontsize=15)
    axs[1].set_ylabel('Omega (rad/s)', fontsize=15)
    axs[1].set_title('Omega moyen {}'.format(commentaire), fontsize=15)
    axs[1].legend()
    plt.tight_layout()
    plt.show()

def plot_strategie(f_array):
    max_vals = f_array.max(axis=0)
    f_flatten = f_array.flatten().copy()
    fig, axs = plt.subplots(1, 2, figsize=(14, 5))
    axs[0].plot(max_vals, label='Strategie maximale')
    axs[0].set_xlabel("Indice de colonne", fontsize=14)
    axs[0].set_ylabel("Valeur maximale", fontsize=14)
    axs[0].set_title("Maximum par colonne", fontsize=16)
    axs[0].legend()
    axs[1].hist(f_flatten[f_flatten > 1e-1], bins=100, color='skyblue', edgecolor=
        'black')
    axs[1].set_xlabel("Valeur (> 0.1)", fontsize=14)
    axs[1].set_ylabel("Nombre d'occurrences", fontsize=14)
    axs[1].set_title("Histogramme des valeurs de la strategie", fontsize=16)
    plt.tight_layout()
    plt.show()

    # --- Recherche de l'equilibre de Nash par point fixe avec ralentissement ---
f_array = np.random.dirichlet(np.ones(N), size=N).T
alphas = np.array([0.1, 0.01, 0.01])
nb_iters = np.array([30, 50, 100])

f_lgn = np.dot(f_array, rho_array)  # Distribution des agents actifs
omega_lgn = np.cumsum(np.array([omega_trapeze(i*dT, (i+1)*dT, f_lgn, C0, Dt, dT, N
    , n_points, D, M) for i in range(N)]))
affichage(f_lgn, omega_lgn, commentaire = "avant optimisation")
plot_strategie(f_array)

for k in range(len(nb_iters)):
    print(f"Phase {k+1}/{len(nb_iters)}: {nb_iters[k]} iterations avec alpha = {
        alphas[k]}")
    nb_iter = nb_iters[k]
    alpha = alphas[k]
    for it in tqdm(range(nb_iter)):
        f_lgn = np.dot(f_array, rho_array)
        omega_lgn = omega_vect(f_lgn, C0, Dt, dT, N, n_points, D, M)
        Couts_integral_array = Couts_integrales(f_lgn, Dt, dT, N, n_points, D, M,
            eps, omega_lgn, omega0)

        for j in range(N):
```

```
            Couts_total_array = Couts_totaux(j, Couts_integral_array)
            i_etoile = np.argmin(Couts_total_array)  # Strategie optimale pour les agents du groupe j
            f_array[:,j] = (1- alpha) *f_array[:,j] + np.array([alpha if i == i_etoile else 0 for i in range(N)])
    affichage(f_lgn, omega_lgn)
    plot_strategie(f_array)
```

C.3 Code en Python pour la sous-section (4.3)

Dans ce code, on choisit T=3600 secondes car on a considéré une stratégie constante dans le temps donc on atteint un régime stationnaire en moins d'une heure. En pratique, il faudrait faire ce code sur T=24*3600 secondes pour obtenir r et Ψ sur toute la journée.

```
#  ----  variables physiques  ----
M = 1e3
P0 = 1e-1
lam = 1
kappa = 1e0
D = 1e-1

#  ----  initialisation  ----
import numpy as np
import matplotlib.pyplot as plt
from numba import njit, prange
from tqdm import tqdm
plt.close('all')

T = 3600
N_t = 1000
dt = T / N_t
N_theta, N_omega = 1000, 1000

print("Pas de discretisation :", dt, "s")
omega_0 = 2*np.pi* 0.05
print("omega_0 :", omega_0, "rad/s")
theta_grid, omega_grid = np.meshgrid(
    np.linspace(-np.pi, np.pi, N_theta, endpoint=False),
    np.linspace(-2/3*omega_0, 2/3*omega_0, N_omega, endpoint=False),
    indexing='ij'
)
dtheta = (theta_grid[-1, 0] - theta_grid[0, 0]) /(N_theta - 1)
domega = (omega_grid[0, -1] - omega_grid[0, 0]) /(N_omega - 1)

# ----  fonctions du code  ----
@njit
def P(t):
    return P0*np.ones((N_theta, N_omega), dtype=np.float64)

@njit(parallel=True)
def demi_evolution_theta(arr):
    new_arr = np.empty_like(arr)
    for i in prange(N_theta):
        for j in range(N_omega):
            i_deplacement = omega_grid[0,j]*((dt/2)/dtheta)
            if i_deplacement >= 0:
                new_arr[i, j] = (1 - i_deplacement%1) * arr[(i - int(i_deplacement))%(N_theta-1), j] + \
                                i_deplacement%1 * arr[(i - int(i_deplacement+1))%(N_theta-1), j]          # la masse qui arrive en i cest celle qui etait en i-v*dt (dt/2 car demi-evolution pour respecter Stang)
            else:
                i_deplacement = -i_deplacement
                new_arr[i, j] =i_deplacement%1 * arr[(i + int(i_deplacement+1))%(N_theta-1), j] + \
                                (1 - i_deplacement%1) * arr[(i + int(i_deplacement))%(N_theta-1), j]
    return new_arr

@njit(parallel=True)
```

```python
def d_domega(arr):
    grad_omega = np.empty_like(arr)
    for i in prange(N_theta):
        for j in range(N_omega):
            if 0 < j < N_omega-1:
                grad_omega[i, j] = (arr[i, j+1] - arr[i, j-1]) / (2*domega)
            elif j == 0:
                grad_omega[i, j] = (arr[i, j+1] - arr[i, j]) / domega
            else:
                grad_omega[i, j] = (arr[i, j] - arr[i, j-1]) / domega
    return grad_omega

@njit(parallel=True)
def drho_dt(rho_t, Pt, r, Psi):
    return -1/M * d_domega( (-kappa*omega_grid + Pt + lam*r*np.sin(Psi - theta_grid)) * rho_t) + \
           2*(D/M)**2 * (d_domega(d_domega(rho_t)) )

@njit(parallel=True)
def runge_kutta_4(i_t, rho_t, r, Psi):
    k1 = drho_dt(rho_t,            P(i_t//N_t),                        r, Psi)
    k2 = drho_dt(rho_t + k1*dt/2, (P(i_t//N_t) + P((i_t+1)//N_t) ) /2, r, Psi)  # Pour evaluer P en t + dt/2, on prend la moyenne des P(t) et P(t+dt/2)
    k3 = drho_dt(rho_t + k2*dt/2, (P(i_t//N_t) + P((i_t+1)//N_t) ) /2, r, Psi)
    k4 = drho_dt(rho_t + k3*dt,    P((i_t+1)//N_t),                    r, Psi)  # Ici on evalue P en t + dt, donc on prend P(t+dt)
    return dt / 6 * (k1 + 2 * k2 + 2 * k3 + k4)

#  ----  boucle et affichage  ----
rho_t = np.exp(-(20*(omega_grid)/omega_0)**2 / 2)*np.exp(-(10*theta_grid/np.pi)**2 / 2)
rho_t /= np.sum(rho_t)
rPsi_list = []
for i_t in tqdm(range(N_t)):
    rPsi_list.append(  np.mean(np.exp(1j*theta_grid[:,:])*rho_t[:,:]) )
    r = np.abs(rPsi_list[-1])
    Psi = np.angle(rPsi_list[-1])
    rho_t = demi_evolution_theta(rho_t)             # Deplacement explicite suivant theta
    rho_t += runge_kutta_4(i_t, rho_t, r, Psi)      # RK4 pour l'evolution en omega
    rho_t[:, 0] = 0.0
    rho_t[:, -1] = 0.0
    rho_t = np.clip(rho_t, 0, None)
    rho_t /= np.sum(rho_t)
    rho_t = demi_evolution_theta(rho_t)             # Deplacement explicite suivant theta
    # Inutile en theorie mais evite erreurs d'arrondis
    rho_t[:, 0] = 0.0
    rho_t[:, -1] = 0.0
    rho_t[rho_t < 0] = 0
    rho_t /= np.sum(rho_t)
    if i_t % (N_t // 10) == 0:
        plt.figure(figsize=(3,3))
        plt.pcolormesh(omega_grid, theta_grid, rho_t, shading='auto', cmap='viridis')
        plt.colorbar(label='Density')
        plt.xlabel('Omega (rad/s)')
        plt.ylabel('Theta (rad)')
        plt.title(f'Temps: {round(i_t*dt/60, 1)} min')
        plt.show()
plt.figure(figsize=(3,3))
plt.pcolormesh(omega_grid, theta_grid, rho_t, shading='auto', cmap='viridis')
plt.colorbar(label='Density')
plt.xlabel('Omega (rad/s)')
plt.ylabel('Theta (rad)')
plt.title(f'Temps: {round(i_t*dt/60, 1)} min')
plt.show()
```

Références

[1] Fonctionnement et réglage des systèmes de transport et de distribution d'électricité en régime permanent. Cours, 2020. CentraleSupélec.

[2] Clémence Alasseur, Imen Ben Tahar, and Anis Matoussi. An extended mean field game for storage in smart grids, 2019.

[3] Philippe-André Martin. Physique statistique des processus irréversibles. Lecture, DEA, 2006.

[4] Francisco A. Rodrigues, Thomas K. D. M. Peron, Peng Ji, and Jürgen Kurths. The kuramoto model in complex networks. *Physics Reports*, 610 :1–98, 2016.

[5] Benjamin Schäfer, Moritz Matthiae, Marc Timme, and Dirk Witthaut. Decentral smart grid control. *New Journal of Physics*, 17(1) :015002, jan 2015.

[6] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3) :506–517, 1968.

[7] Igor Swiecicki, Thierry Gobron, and Denis Ullmo. Schrödinger approach to mean field games. *Phys. Rev. Lett.*, 116 :128701, Mar 2016.

[8] Denis Ullmo, Igor Swiecicki, and Thibault Gobron. Quadratic mean field games. *Physics Reports*, 799 :1–35, 2019.

[9] Jaegon Um, Hyunsuk Hong, and Hyunggyu Park. Validity of annealed approximation in a high-dimensional system. *Scientific Reports*, 14(1), March 2024.